8-Bit Flash MCU with Op Amps & Comparators

HT45F23A

Revision: V1.00 Date: July 03, 2012

www.holtek.com

Table of Contents

Features

CPU Features

- Operating voltage:
  – f_{SYS}= 32.768kHz: 2.2V~5.5V
  – f_{SYS}= 910kHz: 2.2V~5.5V
  – f_{SYS}= 2MHz: 2.2V~5.5V
  – f_{SYS}= 4MHz: 2.2V~5.5V
  – f_{SYS}= 8MHz: 3.3V~5.5V
- TinyPower technology for low power operation
- Power down and wake-up functions to reduce power consumption
- Oscillator types:
  – External Crystal - HXT
  – External 32.768kHz Crystal - LXT
  – External RC - ERC
  – Internal RC - HIRC
  – Internal 32kHz RC - LIRC
- Multi-mode operation: Normal, Slow, Idle and Sleep
- Fully integrated internal 32kHz, 910kHz, 2MHz, 4MHz and 8MHz oscillator requires no external components
- Externally supplied system clock option
- All instructions executed in one or two machine Cycles
- Table read instructions
- 61 or 63 powerful instructions
- 6-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 2K×15
- RAM Data Memory: 128×8
- EEPROM Data Memory: 64×8
- Watchdog Timer function
- Up to 22 bidirectional I/O lines
- Software controlled 4-SCOM lines LCD COM driver with 1/2 bias
- Multiple pin-shared external interrupts
- Single 8-bit programmable Timer/Event Counter with overflow interrupt and Single 16-bit programmable Timer/Event Counter with overflow interrupt function
- Dual Time-Base functions
- Serial Interfaces Module - SIM for SPI or I^2C
- Dual Comparator functions
- Dual Operational Amplifiers functions
- Operational Amplifier output to internal two channel 12-bit ADC function
- Up to 6 channel 12-bit ADC
- Up to 2 channel 8-bit PWM
- 12-bit Audio DAC output
- PFD/Buzzer for audio frequency generation
- Internal 2.4V/3.3V LDO
- Low voltage reset function
- Low voltage detect function
- 16-pin NSOP, 20/24-pin SSOP package types

General Description

The HT45F23A is a Flash Memory Tinypower A/D type 8-bit high performance RISC architecture microcontrollers, designed especially for applications that interface directly to analog signals. The HT45F23A device has a higher gain bandwidth making it more suitable for higher frequency applications.

Offering users the convenience of Flash Memory multi-programming features, the device also includes a wide range of functions and features. Other memory includes an area of RAM Data Memory as well as an area of EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc.

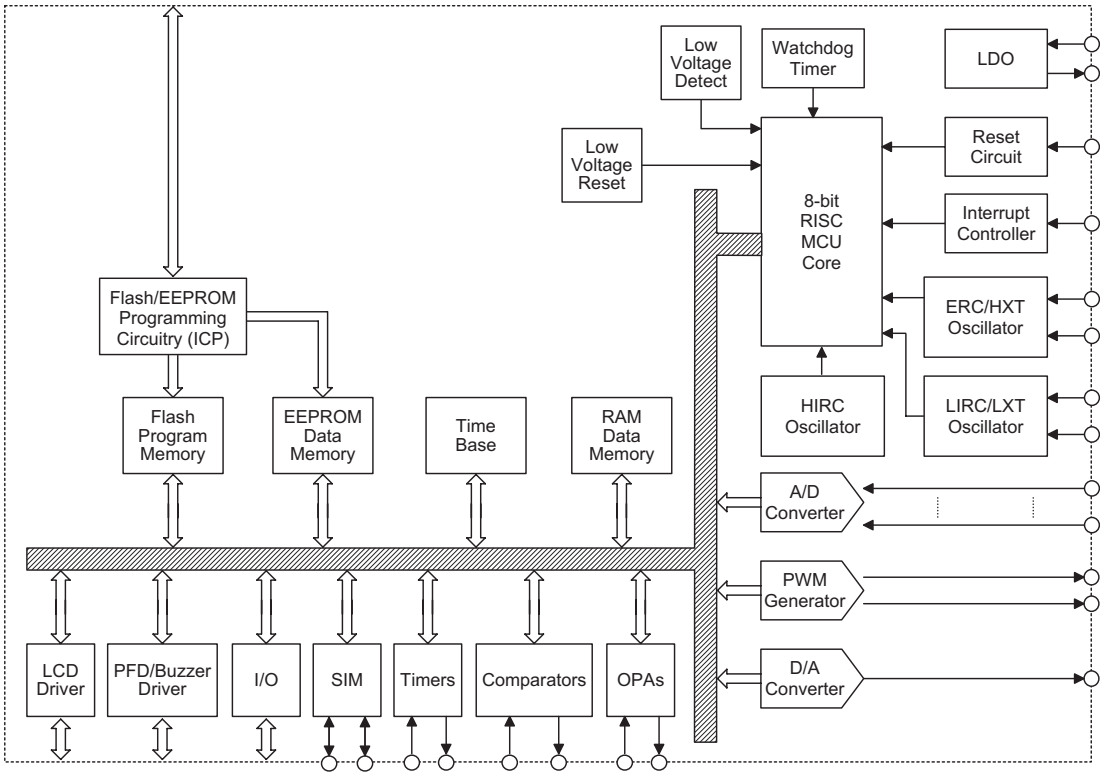
Analog features include an integrated multi-channel Analog to Digital Converter, dual Pulse Width Modulation outputs, dual Operational Amplifiers , dual Comparators , one internal 2.4V or 3.3V LDO (Low Drop Out) for voltage regulator and a 12-bit DAC for voice output application, communication with the outside world is catered for by including fully integrated SPI or I^2C interface functions, two popular interfaces which provide designers with a means of easy communication with external peripheral hardware. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

A full choice of HXT, LXT, ERC, HIRC and LIRC oscillator functions are provided including a fully integrated system oscillator which requires no external components for its implementation. The unique Holtek TinyPower technology also gives the device extremely low current consumption characteristics, an extremely important consideration in the present trend for low power battery powered applications. The usual Holtek MCU features such as power down and wake-up functions, oscillator options, programmable frequency divider, etc. combine to ensure user applications require a minimum of external components.

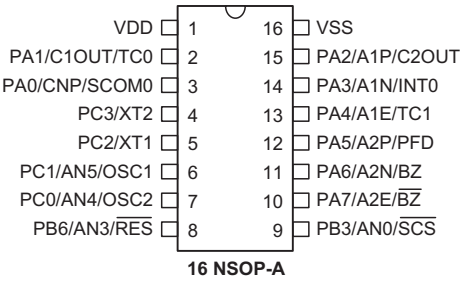
The inclusion of flexible I/O programming features, Time-Base functions along with many other features ensure that the device will find excellent use in applications such as electronic metering, environmental monitoring, handheld instruments, household appliances, electronically controlled tools, motor driving, home security systems related to Smoke detector and many others.

Block Diagram

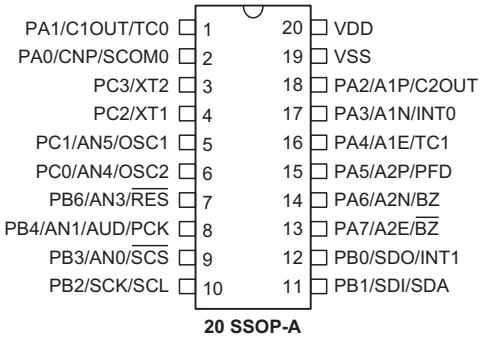
The following block diagram illustrates the main functional blocks.



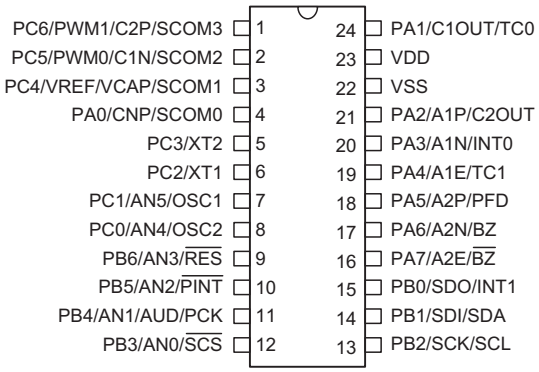
Pin Assignment



16 NSOP-A



20 SSOP-A



24 SSOP-A

Pin Description

Pin Name	Function	OPT	I/T	O/T	Description
PA0/CNP/SCOM0	PA0	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up, wake-up.
	CNP	CMP1C1	CMPI	—	Comparator input pin
	SCOM0	LCDC	—	SCOM	Software controlled 1/2 bias LCD COM
PA1/C1OUT/TC0	PA1	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up, wake-up.
	C1OUT	CMP1C1	—	CMPO	Comparator 1 output pin
	TC0	—	ST	—	External Timer 0 clock input
PA2/A1P/C2OUT	PA2	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up, wake-up.
	A1P	OPA1C1	OPAI	—	OPA1 non-inverting input pin
	C2OUT	CMP2C1	—	CMPO	Comparator 2 output pin
PA3/A1N/INT0	PA3	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up, wake-up.
	A1N	OPA1C1	OPAI	—	OPA1 inverting input pin
	INT0	—	ST	—	External interrupt 0 input pin
PA4/A1E/TC1	PA4	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	A1E	OPA1C1	—	OPAO	OPA1 output pin
	TC1	—	ST	—	External Timer 1 clock input

Pin Name	Function	OPT	I/T	O/T	Description
PA5/A2P/PFD	PA5	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	A2P	OPA2C1	OPAI	—	OPA2 non-inverting input pin
	PFD	MISC	—	CMOS	PFD output
PA6/A2N/BZ	PA6	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	A2N	OPA2C1	OPAI	—	OPA2 inverting input pin
	BZ	BPCTL	—	CMOS	Buzzer output
PA7/A2E/BZ	PA7	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	A2E	OPA2C1	—	OPAO	OPA2 output pin
	BZ	BPCTL	—	CMOS	Complementary buzzer output
PB0/SDO/INT1	PB0	PBPU MISC	ST	CMOS NMOS	General purpose I/O. Register enabled pull-up and output NMOS structure.
	SDO	—	—	CMOS	SPI data output
	INT1	—	ST	—	External interrupt 1 input pin
PB1/SDI/SDA	PB1	PBPU MISC	ST	CMOS NMOS	General purpose I/O. Register enabled pull-up and output NMOS structure.
	SDI	—	ST	—	SPI data input
	SDA	—	ST	NMOS	I^2C data
PB2/SCK/SCL	PB2	PBPU MISC	ST	CMOS NMOS	General purpose I/O. Register enabled pull-up and output NMOS structure.
	SCK	—	ST	—	SPI serial clock
	SCL	—	ST	NMOS	I^2C clock
PB3/AN0/$\overline{SCS}$	PB3	PBPU MISC	ST	CMOS NMOS	General purpose I/O. Register enabled pull-up and output NMOS structure.
	AN0	ADCR	AN	—	A/D channel 0
	$\overline{SCS}$	—	ST	—	SPI slave select
PB4/AN1/AUD/ PCK	PB4	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN1	ADCR	AN	CMOS	A/D channel 1
	AUD	DACTRL	—	—	D/A output pin
	PCK	—	—	CMOS	Peripheral clock output
PB5/AN2/$\overline{PINT}$	PB5	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN2	ADCR	AN	—	A/D channel 2
	$\overline{PINT}$	—	ST	—	Peripheral interrupt
PB6/AN3/$\overline{RES}$	PB6	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN3	ADCR	AN	—	A/D channel 3
	$\overline{RES}$	CO	ST	—	Reset pin
PC0/AN4/ OSC2	PC0	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	AN4	ADCR	AN	—	A/D channel 4
	OSC2	CO	—	HXT	HXT pin
PC1/AN5/ OSC1	PC1	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	AN5	ADCR	AN	—	A/D channel 5
	OSC1	CO	HXT	—	HXT/ERC pin

Pin Name	Function	OPT	I/T	O/T	Description
PC2/XT1	PC2	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	XT1	CO	LXT	—	LXT pin
PC3/XT2	PC3	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	XT2	CO	—	LXT	LXT pin
PC4/VREF/ VCAP/SCOM1	PC4	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	VREF	ACSR	AN	—	ADC reference input
	VCAP	LDOC	—	—	LDO output capacitor pin. Connect a 0.1μF capacitor.
	SCOM1	LCDC	—	SCOM	Software controlled 1/2 bias LCD COM
PC5/PWM0/ C1N/SCOM2	PC5	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PWM0	BPCTL	—	CMOS	PWM0 output pin
	C1N	CMP1C1	CMPI	—	Comparator 1 inverting input pin
	SCOM2	LCDC	—	SCOM	Software controlled 1/2 bias LCD COM
PC6/PWM1/ C2P/SCOM3	PC6	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PWM1	BPCTL	—	CMOS	PWM1 output pin
	C2P	CMP2C1	CMPI	—	Comparator 2 non-inverting input pin
	SCOM3	LCDC	—	SCOM	Software controlled 1/2 bias LCD COM
VDD	VDD	—	PWR	—	Power supply
VSS	VSS	—	PWR	—	Ground

Note: I/T: Input type
O/T: Output type
OPT: Optional by configuration option (CO) or register option
PWR: Power
CO: Configuration option
ST: Schmitt Trigger input
CMOS: CMOS output
SCOM: Software controlled LCD COM
HXT: High frequency crystal oscillator
LXT: Low frequency crystal oscillator
OPAI: Operational Amplifier input
OPAO: Operational Amplifier output
CMPI: Comparator input
CMPO: Comparator output
DAO: D/A output

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}−0.3V to V_{SS}+6.0V
Input Voltage ..... V_{SS}−0.3V to V_{DD}+0.3V
Storage Temperature..... -50˚C to 125˚C
Operating Temperature..... -40˚C to 85˚C
I_{OH} Total ..... -100mA
I_{OL} Total 100mA
Total Power Dissipation 500mW

Note: These are stress ratings only. Stresses exceeding the range specified under "Absolute Maximum Ratings" may cause substantial damage to these devices. Functional operation of these devices at other conditions beyond those listed in the specification is not implied and prolonged exposure to extreme conditions may affect devices reliability.

D.C. Characteristics

Ta= 25˚C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	f_{SYS}=910kHz, (HXT/ERC/HIRC)	2.2	—	5.5	V
			f_{SYS}=2MHz, (HXT/ERC/HIRC)	2.2	—	5.5	V
			f_{SYS}=4MHz, (HXT/ERC/HIRC/EC)	2.2	—	5.5	V
			f_{SYS}=8MHz (HXT/ERC/HIRC/EC)	3.3	—	5.5	V
I_{DD1}	Operating Current (HXT, ERC)	3.3V	No load, f_{SYS}=f_M=455kHz, ADC off, LVR off, Comparator off, OPAs off	—	70	110	μA
	Operating Current (HXT, ERC)		No load, f_{SYS}=f_M= 455kHz, ADC off, LVR on, Comparator on, OPAs off	—	100	150	μA
I_{DD2}	Operating Current (ERC, HIRC)	3.3V	No load, f_M=910kHz, f_{SYS}=f_{SLOW}=455kHz, ADC off, LVR off, Comparator off, OPAs off	—	90	135	μA
	Operating Current (ERC, HIRC)		No load, f_M=910kHz, f_{SYS}=f_{SLOW}=455kHz, ADC off, LVR on, Comparator on, OPAs off	—	120	180	μA
I_{DD3}	Operating Current (ERC, HIRC)	3.3V	No load, f_{SYS}=f_M=910kHz, ADC off, LVR off, Comparator off, OPAs off	—	110	170	μA
	Operating Current (ERC, HIRC)		No load, f_{SYS}=f_M=910kHz, ADC off, LVR on, Comparator on, OPAs off	—	160	240	μA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD4}	Operating Current (HXT, ERC)	3.3V	No load, $f_{SYS}=f_M$=1MHz, ADC off, LVR off, Comparator off, OPAs off	—	120	180	μA
	Operating Current (HXT, ERC)		No load, $f_{SYS}=f_M$=1MHz, ADC off, LVR on, Comparator on, OPAs off	—	170	260	μA
I_{DD5}	Operating Current (HXT, ERC, HIRC)	3.3V	No load, $f_{SYS}=f_M$=2MHz, ADC off, LVR off, Comparator off, OPAs off	—	170	260	μA
	Operating Current (HXT, ERC, HIRC)		No load, $f_{SYS}=f_M$=2MHz, ADC off, LVR on, Comparator on, OPAs off	—	200	300	μA
I_{DD6}	Operating Current (HXT, ERC, HIRC)	3V	No load, $f_{SYS}=f_M$=4MHz, ADC off	—	420	630	μA
		5V		—	700	1000	μA
I_{DD7}	Operating Current (HXT, ERC, HIRC)	5V	No load, $f_{SYS}=f_M$=8MHz, ADC off	—	1.5	3.0	mA
I_{DD9}	Operating Current (Slow Mode, f_M=4MHz) (HXT, ERC, HIRC)	3V	No load, $f_{SYS}=f_{SLOW}$=1MHz, ADC off	—	200	300	μA
		5V		—	400	600	μA
I_{DD10}	Operating Current (Slow Mode, f_M=4MHz) (HXT, ERC, HIRC)	3V	No load, $f_{SYS}=f_{SLOW}$=2MHz, ADC off	—	250	375	μA
		5V		—	560	840	μA
I_{DD11}	Operating Current (Slow Mode, f_M=8MHz) (HXT, ERC, HIRC)	3V	No load, $f_{SYS}=f_{SLOW}$=2MHz, ADC off	—	300	450	μA
		5V		—	680	1020	μA
I_{DD12}	Operating Current (Slow Mode, f_M=8MHz) (HXT, ERC, HIRC)	3V	No load, $f_{SYS}=f_{SLOW}$=4MHz, ADC off	—	450	800	μA
		5V		—	1000	1500	μA
I_{DD13}	Operating Current (f_{SYS}=LXT (note 1) or LIRC)	3V	No load, WDT off, ADC off	—	10	20	μA
		5V		—	20	35	μA
I_{STB1}	Standby Current (Sleep) (f_{SYS}, f_{SUB}, f_S, f_{WDT}=off)	3V	No load, system HALT, WDT off	—	0.1	1.0	μA
		5V		—	0.2	2.0	μA
I_{STB2}	Standby Current (Sleep) (f_{SYS} Off; f_S On; $f_{WDT}=f_{SUB}$=LXT (note 1) or LIRC)	3V	No load, system HALT, WDT on	—	2	4	μA
		5V		—	4	6	μA
I_{STB3}	Standby Current (Idle) (f_{SYS} Off; f_{WDT} Off; $f_S=f_{SUB}$=LXT (note 1) or LIRC)	3V	No load, system HALT, WDT off	—	4	6	μA
		5V		—	6	9	μA
I_{STB4}	Standby Current (Idle) (f_{SYS} On, $f_{SYS}=f_M$=4MHz; f_{WDT} Off; $f_S=f_{SUB}$=LXT (note 1) or LIRC)	3V	No load, system HALT, WDT off, SPI or I^2C on, PCLK on, PCLK=f_{SYS}/8	—	260	350	μA
		5V		—	350	660	μA
V_{IL1}	Input Low Voltage for I/O, TMRn and INTn	—	—	0	—	0.3V_{DD}	V
V_{IH1}	Input High Voltage for I/O, TMRn and INTn	—	—	0.7V_{DD}	—	V_{DD}	V
V_{IL2}	Input Low Voltage ($\overline{RES}$)	—	—	0	—	0.4V_{DD}	V
V_{IH2}	Input High Voltage ($\overline{RES}$)	—	—	0.9V_{DD}	—	V_{DD}	V
V_{IL3}	Input Low Voltage (PB1~PB3)	5V	—	—	—	1	V
V_{IH3}	Input High Voltage (PB1~PB3)	5V	—	2	—	—	V
V_{LVR1}	Low Voltage Reset	—	V_{LVR}=2.10V	-5%× Typ.	2.10	+5%× Typ.	V
V_{LVR2}			V_{LVR}=2.55V		2.55		
V_{LVR3}			V_{LVR}=3.15V		3.15		
V_{LVR4}			V_{LVR}=4.20V		4.20		

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVD1}	Low Voltage Detector Voltage	—	LVDEN=1, V_{LVD}=2.0V	-5%× Typ.	2.0	+5%× Typ.	V
V_{LVD2}			LVDEN=1, V_{LVD}=2.2V		2.2		
V_{LVD3}			LVDEN=1, V_{LVD}=2.4V		2.4		
V_{LVD4}			LVDEN=1, V_{LVD}=2.7V		2.7		
V_{LVD5}			LVDEN=1, V_{LVD}=3.0V		3.0		
V_{LVD6}			LVDEN=1, V_{LVD}=3.3V		3.3		
V_{LVD7}			LVDEN=1, V_{LVD}=3.6V		3.6		
V_{LVD8}			LVDEN=1, V_{LVD}=4.4V		4.4		
I_{OL}	I/O Port Sink Current	3V	V_{OL}=0.1V_{DD}	6	12	—	mA
		5V		10	25	—	mA
I_{OH}	I/O Port Source Current	3V	V_{OH}=0.9V_{DD}	-2	-4	—	mA
		5V		-5	-8	—	mA
R_{PH}	Pull-high Resistance	3V	—	40	60	80	kΩ
		5V	—	10	30	50	kΩ
AV_{DD}	A/D Converter Operating Voltage	—	—	2.7	—	5.5	V
V_{AD}	A/D Input Voltage	—	—	0	—	V_{REF}	V
V_{REF}	A/D Input Reference Voltage Range	—	AV_{DD}=5V	2	—	V_{DD}	V
DNL	ADC Differential Non-Linearity	3V	V_{REF}=V_{DD}, t_{AD}=1μs	—	±1	±2	LSB
		5V					
INL	ADC Integral Non-Linearity	3V	V_{REF}=V_{DD}, t_{AD}=1μs	—	±2	±4	LSB
		5V					
I_{ADC}	Additional Power Consumption if A/D Converter is Used	3V	—	—	0.5	1.0	mA
		5V	—	—	1.5	3.0	mA
V_{BG}	Bandgap Reference with Buffer voltage	—	—	-3%	1.25	+3%	V
I_{LVR}	DC current when LVR or LVD turn on	3V	—	—	10	15	μA
		5V		—	20	30	μA
V_{SCOM}	V_{DD}/2 Voltage for LCD COM	5V	No load	0.475	0.500	0.525	VDD
	V_{LDO}/2 Voltage for LCD COM	5V	No load	0.475	0.500	0.525	VLDO
I_{DD}	Quiescent current	5V	No load, A1OEN/A2OEN fixed to 0	—	120	—	μA
I_{OUT}	Output Current	5V	ISEL=0, LCDBUF=disable	—	10	—	μA
			ISEL=1, LCDBUF=disable	—	25	—	μA
			ISEL=0, LCDBUF=enable	—	2	—	mA
			ISEL=1, LCDBUF=enable	—	2	—	mA

Note: 1. t_{SYS}= 1/f_{SYS}; t_{SUB} = 1/f_{SUB}

2. Detailed showing please refer LDO section.

A.C. Characteristics

Ta= 25˚C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS1}	System Clock (HXT, ERC, HIRC)	—	2.2V~5.5V	400	—	4000	kHz
			3.3V~5.5V	400	—	8000	kHz
f_{SYS2}	8MHz HIRC	3.3V	Ta=25˚C	-2%	8	+2%	MHz
		3.3V	Ta=-40˚C~85˚C	-5%	8	+5%	MHz
		2.7V~5.5V	Ta=-40˚C~85˚C	-10%	8	+10%	MHz
f_{SYS3}	4MHz HIRC	3.3V	Ta=25˚C	-2%	4	+2%	MHz
		3.3V	Ta=-40˚C~85˚C	-5%	4	+5%	MHz
		2.7V~5.5V	Ta=-40˚C~85˚C	-10%	4	+10%	MHz
f_{SYS4}	2MHz HIRC	3.3V	Ta=25˚C	-2%	2	+2%	MHz
		3.3V	Ta=-40˚C~85˚C	-5%	2	+5%	MHz
		2.7V~5.5V	Ta=-40˚C~85˚C	-10%	2	+10%	MHz
f_{SYS5}	910kHz HIRC	3.3V	Ta=25˚C	-2%	0.91	+2%	MHz
		3.3V	Ta=-40˚C~85˚C	-5%	0.91	+5%	MHz
		2.7V~5.5V	Ta=-40˚C~85˚C	-10%	0.91	+10%	MHz
f_{LXT}	System Clock (LXT)	—	—	—	32.768	—	kHz
f_{ERC}	4MHz ERC (Note 3)	3.3V	R=150kΩ, Ta=25˚C	-2%	4	+2%	MHz
		3.3V	R=150kΩ, Ta=-40˚C~85˚C	-8%	4	+8%	MHz
		2.7V~5.5V	R=150kΩ, Ta=-40˚C~85˚C	-15%	4	+15%	MHz
t_{LIRC}	32kHz RC Period	3V	—	28.10	31.25	34.40	μs
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{LVR}	Low Voltage width to Reset	—	—	60	120	240	μs
t_{LVD}	Low Voltage width to Interrupt	—	—	1	—	2	t_{SUB}
t_{LVDS}	LVDO Stable Time	5V	LVR disable, LVD enable, VBG is ready	—	—	100	μs
t_{SST1}	System start-up timer period (W/O fast start-up) of HXT/TBC	—	Power up or wake-up from Sleep mode	—	1024	—	t_{SYS}* (Note 1)
t_{SST2}	System start-up timer period of ERC, HIRC, EC	—	Power up or wake-up from HALT (Idle or Sleep mode)	—	1	2	t_{SYS}
t_{SST3}	System start-up timer period (With fast start-up) of HXT/TBC	—	wake-up from Idle mode (f_{SL}=f_{TBC})	—	1	2	t_{TBC} (Note 2)
t_{SST4}	System start-up timer period (With fast start-up) of HXT/TBC	—	wake-up from Idle mode (f_{SL}=f_{LIRC})	—	1	2	t_{LIRC}
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs
t_{AD}	A/D Clock Period	—	—	0.5	—	—	μs
t_{ADC}	A/D Conversion Time (Note 4)	—	—	—	16	—	t_{AD}
t_{ON2ST}	A/D on to A/D Start	—	2.2V~5.5V	2	—	—	μs

Note: 1. t_{SYS}=1/f_{SYS}; t_{SUB}=1/f_{SUB}

2. t_{TBC} is period of LXT or LIRC and it will not stop at Idle mode.

3. For f_{ERC}, as the resistor tolerance will influence the frequency a precision resistor is recommended.

4. ADC conversion time (t_{AD})=n (bits ADC) + 4 (sampling time). The conversion for each bit needs one ADC clock (t_{AD}).

OP Amplifier Electrical Characteristics

Ta= 25˚C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
D.C. Characteristic							
V_{DD}	Operating voltage	—	—	2.2	—	5.5	V
I_{DD}	Quiescent current	5V	No load, A1OEN/A2OEN fixed to 0	—	200	350	μA
V_{OPOS}	Input offset voltage	5V	—	-1	—	+1	mV
I_{OPOS}	Input offset current	—	V_{DD}=5V, V_{CM}=1/2V_{DD}, Ta=40˚C~85˚C	—	10	—	nA
V_{CM}	Common Mode Voltage Range	—	—	V_{SS}	—	V_{DD}-1.4	V
PSRR	Power Supply Rejection Ratio	—	—	58	80	—	dB
CMRR	Common Mode Rejection Ratio	—	V_{DD}=5V V_{CM}=0~V_{DD}-1.4V	58	80	—	dB
A.C. Characteristic							
A_{OL}	Open Loop Gain	—	—	60	80	—	dB
SR	Slew Rate+, Rate-	—	No load	—	0.01	—	V/μs
GBW	Gain Band Width	—	R_L=1MΩ, C_L=100pF	100k	3.5M	—	Hz

Comparator Electrical Characteristics

Ta= 25˚C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DDC}	Comparator Operating Voltage	—	—	2.2	—	5.5	V
I_{DDC}	Comparator Operating Current	3V	—	—	20	40	μA
		5V		—	30	60	μA
V_{CPOS1}	Comparator Input Offset Voltage	5V	CxOF4~0=(10000)	-10	—	+10	mV
V_{CPOS2}	Comparator Input Offset Voltage	5V	By calibration	-4	—	+4	mV
V_{CM}	Comparator Common Mode Voltage Range	—	—	V_{SS}	—	V_{DD}-1.4	V
AOL	Comparator Open Loop Gain	—	—	60	80	—	dB
t_{PD1}	Comparator Response Time	—	With 2mV overdrive	—	—	2	μs
t_{PD2}	Comparator Response Time	—	With 10mV overdrive	—	—	1.5	μs

LDO 2.4V

Ta= 25˚C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DDIN}	Supply Voltage	—	—	2.7	—	5.5	V
V_{DDOUT}	Output Voltage	—	—	2.28	2.40	2.52	V
I_{DD}	Current Consumption	—	After startup, no load	—	50	70	μA
I_{OUT}	Output Current	5V	V_{CAP}=1μF	200	—	1100	μA

LDO 3.3V

Ta= 25°C

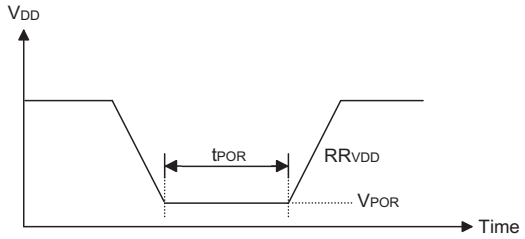
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DDIN}	Supply Voltage	—	—	3.6	—	5.5	V
V_{DDOUT}	Output Voltage	—	—	3.13	3.30	3.46	V
I_{DD}	Current Consumption	—	After startup, no load	—	50	70	μA
I_{OUT}	Output Current	5V	V_{CAP}=1μF	200	—	1100	μA

Note: 1.This LDO can provide stable power supply for PIR sensor with a 10μF cap.
2. The VREF pin should be connected to 10μF for ADC reference voltage and 10μF for PIR sensor.

Power-on Reset Characteristics

Ta= 25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{POR}	VDD Start Voltage to Ensure Power-on Reset	—	—	—	—	100	mV
RR_{VDD}	VDD raising rate to Ensure Power-on Reset	—	—	0.035	—	—	V/ms
t_{POR}	Minimum Time for VDD Stays at V_{POR} to Ensure Power-on Reset	—	—	1	—	—	ms



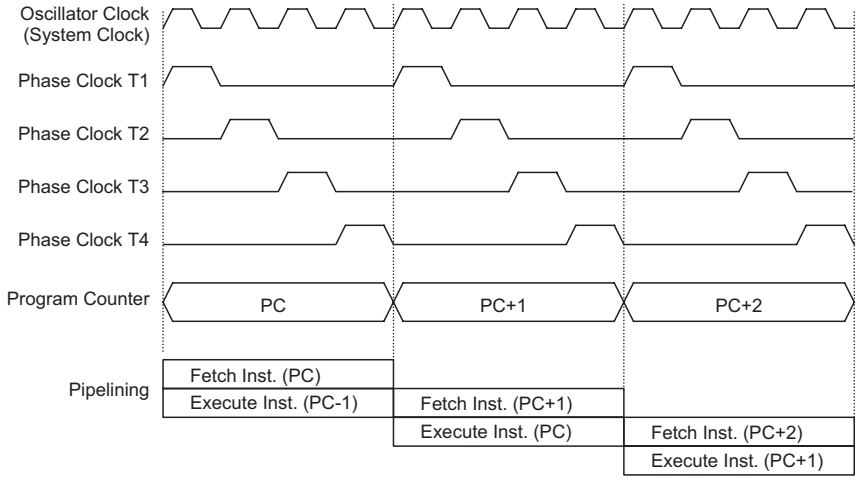
System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to the internal system architecture. The device takes advantage of the usual features found within RISC microcontrollers providing increased speed of operation and enhanced performance. The pipelining scheme is implemented in such a way that instruction fetching and instruction execution are overlapped, hence instructions are effectively executed in one cycle, with the exception of branch or call instructions. An 8-bit wide ALU is used in practically all operations of the instruction set. It carries out arithmetic operations, logic operations, rotation, increment, decrement, branch decisions, etc. The internal data path is simplified by moving data through the Accumulator and the ALU. Certain internal registers are implemented in the Data Memory and can be directly or indirectly addressed. The simple addressing methods of these registers along with additional architectural features ensure that a minimum of external components is required to provide a functional I/O and A/D control system with maximum reliability and flexibility. This makes the device suitable for low-cost, high-volume production for controller applications.

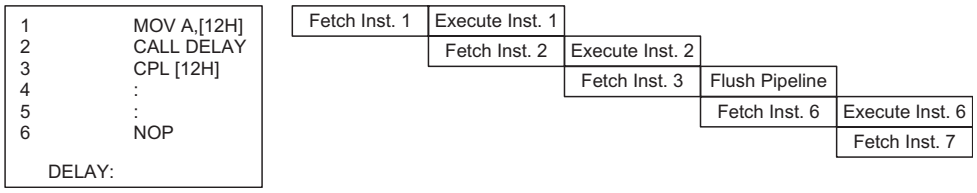
Clocking and Pipelining

The main system clock, derived from either a Crystal/ Resonator or RC oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4 clocks carry out the decoding and execution functions. In this way, one T1~T4 clock cycle forms one instruction cycle. Although the fetching and execution of instructions takes place in consecutive instruction cycles, the pipelining structure of the microcontroller ensures that instructions are effectively executed in one instruction cycle. The exception to this are instructions where the contents of the Program Counter are changed, such as subroutine calls or jumps, in which case the instruction will take one more instruction cycle to execute.

For instructions involving branches, such as jump or call instructions, two instruction cycles are required to complete instruction execution. An extra cycle is required as the program takes one cycle to first obtain the actual jump or call address and then another cycle to actually execute the branch. The requirement for this extra cycle should be taken into account by programmers in timing sensitive applications.



System Clocking and Pipelining



Instruction Fetching

Program Counter

During program execution, the Program Counter is used to keep track of the address of the next instruction to be executed. It is automatically incremented by one each time an instruction is executed except for instructions, such as "JMP" or "CALL" that demand a jump to a non-consecutive Program Memory address. Note that the Program Counter width varies with the Program Memory capacity depending upon which device is selected. However, it must be noted that only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by the application program.

When executing instructions requiring jumps to non-consecutive addresses such as a jump instruction, a subroutine call, interrupt or reset, etc., the microcontroller manages program control by loading the required address into the Program Counter. For conditional skip instructions, once the condition has been met, the next instruction, which has already been fetched during the present instruction execution, is discarded and a dummy cycle takes its place while the correct instruction is obtained.

Program Counter	
Program Counter High Byte	**PCL Register**
PC10~PC8	PCL7~PCL0

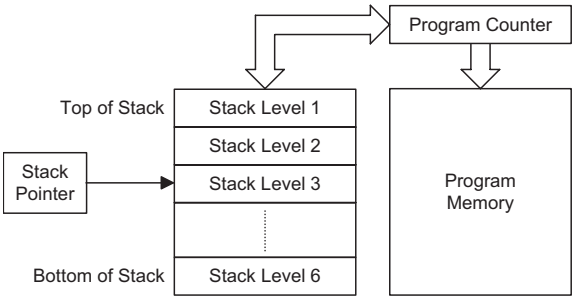
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly. However, as only this low byte is available for manipulation, the jumps are limited to the present page of memory that is 256 locations. When such program jumps are executed, it should also be noted that a dummy cycle will be inserted.

The lower byte of the Program Counter is fully accessible under program control. Manipulating the PCL might cause program branching, so an extra cycle is needed to pre-fetch. Further information on the PCL register can be found in the Special Function Register section.

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organized into 6 levels and is neither part of the Data or Program Memory space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, SP, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.

If the stack is full and an enabled interrupt takes place, the interrupt request flag will be recorded but the acknowledge signal will be inhibited. When the Stack Pointer is decremented, by RET or RETI, the interrupt will be serviced. This feature prevents stack overflow allowing the programmer to use the structure more easily. However, when the stack is full, a CALL subroutine instruction can still be executed which will result in a stack overflow. Precautions should be taken to avoid such cases which might cause unpredictable program branching.



Arithmetic and Logic Unit – ALU

The arithmetic-logic unit or ALU is a critical area of the microcontroller that carries out arithmetic and logic operations of the instruction set. Connected to the main microcontroller data bus, the ALU receives related instruction codes and performs the required arithmetic or logical operations after which the result will be placed in the specified register. As these ALU calculation or operations may result in carry, borrow or other status changes, the status register will be correspondingly updated to reflect these changes. The ALU supports the following functions:

- Arithmetic operations: ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations: AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement INCA, INC, DECA, DEC
- Branch decision, JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI

Flash Program Memory

The Program Memory is the location where the user code or program is stored. For this device the Program Memory is Flash type, which means it can be programmed and re-programmed a large number of times, allowing the user the convenience of code modification on the same device. By using the appropriate programming tools, this Flash device offer users the flexibility to conveniently debug and develop their applications while also offering a means of field programming and updating.

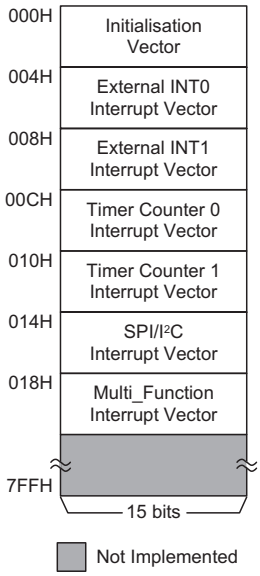
Structure

The Program Memory has a capacity of 2K×15 . The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by separate table pointer registers.

Special Vectors

Within the Program Memory, certain locations are reserved for special usage such as reset and interrupts.

- Location 000H
  This vector is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.
- Location 004H
  This vector is used by the external interrupt 0. If the external interrupt pin receives an active edge, the program will jump to this location and begin execution if the external interrupt is enabled and the stack is not full.
- Location 008H
  This vector is used by the external interrupt 1. If the external interrupt pin receives an active edge, the program will jump to this location and begin execution if the external interrupt is enabled and the stack is not full.
- Location 00CH
  This internal vector is used by the Timer/Event Counter 0. If a Timer/Event Counter 0 overflow occurs, the program will jump to this location and begin execution if the timer/event counter interrupt is enabled and the stack is not full.
- Location 010H
  This internal vector is used by the Timer/Event Counter 1. If a Timer/Event Counter 1 overflow occurs, the program will jump to this location and begin execution if the timer/event counter interrupt is enabled and the stack is not full.
- Location 014H
  This internal vector is used by the SPI/I^2C interrupt. When either an SPI or I^2C bus, dependent upon which one is selected, requires data transfer, the program will jump to this location and begin execution if the SPI/I^2C interrupt is enabled and the stack is not full.
- Location 018H
  This internal vector is used by the Multi-function Interrupt. When the Time Base overflows, the A/D converter completes its conversion process, an active edge appears on the External Peripheral interrupt pin, a Comparator output interrupt, an EEPROM Write or Read cycle ends interrupt, or a LVD detection interrupt, the program will jump to this location and begin execution if the relevant interrupt is enabled and the stack is not full.



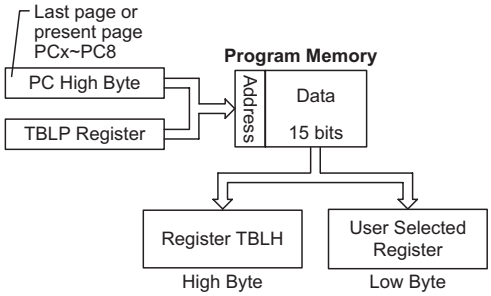
Program Memory Structure

Look-up Table

Any location within the Program Memory can be defined as a look-up table where programmers can store fixed data. To use the look-up table, the table pointer must first be setup by placing the lower order address of the look up data to be retrieved in the table pointer register, TBLP. This register defines the lower 8-bit address of the look-up table.

After setting up the table pointer, the table data can be retrieved from the current Program Memory page or last Program Memory page using the "TABRDC[m]" or "TABRDL [m]" instructions, respectively. When these instructions are executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register. Any unused bits in this transferred higher order byte will be read as "0".

The following diagram illustrates the addressing/data flow of the look-up table:



Instruction	Table Location Bits										
	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TABRDC[m]	PC10	PC9	PC8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL[m]	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

Table Location

Note: PC10~PC8:Current Program Counter bits

@7~@0:Table Pointer TBLP bits

Table Program Example

The accompanying example shows how the table pointer and table data is defined and retrieved from the device. This example uses raw table data located in the last page which is stored there using the ORG statement. The value at this ORG statement is "0700H" which refers to the start address of the last page within the 2K Program Memory of the device. The table pointer is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "0706H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address of the present page if the "TABRDC [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRDL [m]" instruction is executed.

Because the TBLH register is a read-only register and cannot be restored, care should be taken to ensure its protection if both the main routine and Interrupt Service Routine use the table read instructions. If using the table read instructions, the Interrupt Service Routines may change the value of TBLH and subsequently cause errors if used again by the main routine. As a rule it is recommended that simultaneous use of the table read instructions should be avoided. However, in situations where simultaneous use cannot be avoided, the interrupts should be disabled prior to the execution of any main routine table-read instructions. Note that all table related instructions require two instruction cycles to complete their operation.

Table Read Program Example

```
tempreg1  db  ?    ; temporary register #1
tempreg2  db  ?    ; temporary register #2
     :
     :
mov  a,06h         ; initialise table pointer - note that this address is referenced
mov  tblp,a        ; to the last page or present page
     :
     :
tabrdl  tempreg1   ; transfers value in table referenced by table pointer to tempregl
                   ; data at prog. memory address "0706H" transferred to tempreg1 and TBLH
dec  tblp          ; reduce value of table pointer by one

tabrdl  tempreg2   ; transfers value in table referenced by table pointer to tempreg2
                   ; data at prog.memory address "0705H" transferred to tempreg2 and TBLH
                   ; in this example the data "1AH" is transferred to tempreg1 and data "0FH"
                   ; to register tempreg2 the value "00H" will be transferred to the high
                   ; byte register TBLH
        :
        :
org  700h          ; sets initial address of last page
dc   00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
        :
        :
```

In Circuit Programming

The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 5-pin interface. This provides manufacturers with the possibility of manufacturing their circuit boards complete with a programmed or un-programmed microcontroller, and then programming or upgrading the program at a later stage. This enables product manufacturers to easily keep their manufactured products supplied with the latest program releases without removal and re-insertion of the device.

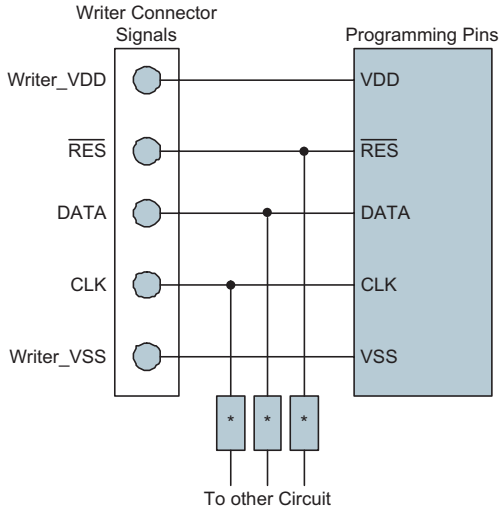
Programming Pins	Function
DATA	Serial Data Input/Output
CLK	Serial Clock
$\overline{RES}$	Device Reset
VDD	Power Supply
VSS	Ground

The Program Memory and EEPROM data memory can both be programmed serially in-circuit using this 5-wire interface. Data is downloaded and uploaded serially on a single pin with an additional line for the clock. Two additional lines are required for the power supply and one line for the reset. The technical details regarding the in-circuit programming of the device are beyond the scope of this document and will be supplied in supplementary literature.

During the programming process the $\overline{RES}$ pin will be held low by the programmer disabling the normal operation of the microcontroller and taking control of the PA0 and PA2 I/O pins for data and clock programming purposes. The user must there take care to ensure that no other outputs are connected to these two pins.

Programmer Pin	MCU Pins
$\overline{RES}$	PB6
DATA	PA0
CLK	PA2

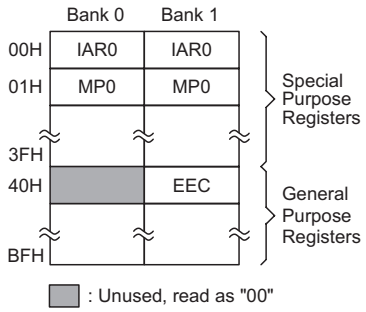
Programmer and MCU Pins



Note: * may be resistor or capacitor. The resistance of * must be greater than 1kΩ or the capacitance of * must be less than 1nF.

RAM Data Memory

The Data Memory is a volatile area of 8-bit wide RAM internal memory and is the location where temporary information is stored.



Data Memory Structure

Note: Most of the Data Memory bits can be directly manipulated using the"SET [m].i" and "CLR [m].i" with the exception of a few dedicated bits. The Data Memory can also be accessed through the memory pointer registers.

Structure

Divided into two sections, the first of these is an area of RAM where special function registers are located. These registers have fixed locations and are necessary for correct operation of the device. Many of these registers can be read from and written to directly under program control, however, some remain protected from user manipulation. The second area of Data Memory is reserved for general purpose use. All locations within this area are read and write accessible under program control.

The overall Data Memory is subdivided into two banks. The Special Purpose Data Memory registers are accessible in all banks, with the exception of the EEC register at address 40H, which is only accessible in Bank 1. Switching between the different Data Memory banks is achieved by setting the Bank Pointer to the correct value. The start address of the Data Memory is the address "00H".

All microcontroller programs require an area of read/write memory where temporary data can be stored and retrieved for use later. It is this area of RAM memory that is known as General Purpose Data Memory. This area of Data Memory is fully accessible by the user program for both read and write operations. By using the "SET [m].i" and "CLR [m].i" instructions individual bits can be set or reset under program control giving the user a large range of flexibility for bit manipulation in the Data Memory.

Special Function Registers

Most of the Special Function Register details will be described in the relevant functional section, however several registers require a separate description in this section.

	Bank 0, 1
00H	IAR0
01H	MP0
02H	IAR1
03H	MP1
04H	BP
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	TBC
0AH	STATUS
0BH	INTC0
0CH	Unused
0DH	TMR0
0EH	TMR0C
0FH	TMR1H
10H	TMR1L
11H	TMR1C
12H	PA
13H	PAC
14H	PB
15H	PBC
16H	PC
17H	PCC
18H	LVDC
19H	DACTRL
1AH	PWM0
1BH	PWM1
1CH	BPCTL
1DH	WDTC
1EH	INTC1
1FH	OPA1C0
20H	OPA1C1

	Bank 0	Bank 1
21H	OPA2C0	
22H	OPA2C1	
23H	OPA2C2	
24H	ADRL	
25H	ADRH	
26H	ADCR	
27H	ACSR	
28H	SMOD	
29H	PAWU	
2AH	PAPU	
2BH	PBPU	
2CH	PCPU	
2DH	ADPCR	
2EH	INTEDGE	
2FH	CMP1C0	
30H	CMP1C1	
31H	CMP2C0	
32H	CMP2C1	
33H	MISC	
34H	MFIC0	
35H	MFIC1	
36H	SIMC0	
37H	SIMC1	
38H	SIMD	
39H	SIMA/SIMC2	
3AH	EEA	
3BH	EED	
3CH	LCDC	
3DH	LDOC	
3EH	DAL	
3FH	DAH	
40H	Unused	EEC

Special Purpose Data Memory

Indirect Addressing Registers – IAR0, IAR1

The Indirect Addressing Registers, IAR0 and IAR1, although having their locations in normal RAM register space, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses these Indirect Addressing Registers and Memory Pointers, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0 and IAR1 registers will result in no actual read or write operation to these registers but rather to the memory location specified by their corresponding Memory Pointer, MP0 or MP1. Acting as a pair, IAR0 with MP0 and IAR1 with MP1, can together access data from the Data Memory. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers indirectly will return a result of "00H" and writing to the registers indirectly will result in no operation.

Memory Pointers – MP0, MP1

Two Memory Pointers, known as MP0 and MP1 are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the microcontroller is directed to, is the address specified by the related Memory Pointer. MP0, together with Indirect Addressing Register, IAR0, are used to access data from Bank 0, while MP1 and IAR1 are used to access data from all banks according to BP register. Direct Addressing can only be used with Bank 0, all other Banks must be addressed indirectly using MP1 and IAR1.

The following example shows how to clear a section of four Data Memory locations already defined as locations adres1 to adres4.

Indirect Addressing Program Example

```
data .section  "data"
adres1 db ?
adres2 db ?
adres3 db ?
adres4 db ?
block  db ?
code .section at 0 code
org    00h

start:
mov a,04h                ; setup size of block
mov block,a
mov a,offset adres1      ; Accumulator loaded with first RAM address
mov mp0,a                ; setup memory pointer with first RAM address

loop:
clr IAR0                 ; clear the data at address defined by MP0
inc mp0                  ; increment memory pointer
sdz block                ; check if last memory location has been cleared
jmp loop

continue:
```

The important point to note here is that in the example shown above, no reference is made to specific Data Memory addresses.

Bank Pointer BP

The Data Memory is divided into two Banks, known as Bank 0 and Bank 1. A Bank Pointer, which is bit 0 of the Bank Pointer register is used to select the required Data Memory bank. Only data in Bank 0 can be directly addressed as data in Bank 1 must be indirectly addressed using Memory Pointer MP1 and Indirect Addressing Register IAR1. Using Memory Pointer MP0 and Indirect Addressing Register IAR0 will always access data from Bank 0, irrespective of the value of the Bank Pointer. Memory Pointer MP1 and Indirect Addressing Register IAR1 can indirectly address data in either Bank 0 or Bank 1 depending upon the value of the Bank Pointer.

The Data Memory is initialised to Bank 0 after a reset, except for the WDT time-out reset in the Idle/Sleep Mode, in which case, the Data Memory bank remains unaffected. It should be noted that Special Function Data Memory is not affected by the bank selection, which means that the Special Function Registers can be accessed from within either Bank 0 or Bank 1. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of the Bank Pointer.

BP Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	DMBP0
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7 ~ 1 Unimplemented, read as "0"

Bit 0 **DMBP0**: Select Data Memory Banks
0: Bank 0
1: Bank 1

Accumulator – ACC

The Accumulator is central to the operation of any microcontroller and is closely related with operations carried out by the ALU. The Accumulator is the place where all intermediate results from the ALU are stored. Without the Accumulator it would be necessary to write the result of each calculation or logical operation such as addition, subtraction, shift, etc., to the Data Memory resulting in higher programming and timing overheads. Data transfer operations usually involve the temporary storage function of the Accumulator; for example, when transferring data between one user defined register and another, it is necessary to do this by passing the data through the Accumulator as no direct transfer between two registers is permitted.

Program Counter Low Register – PCL

To provide additional program control functions, the low byte of the Program Counter is made accessible to programmers by locating it within the Special Purpose area of the Data Memory. By manipulating this register, direct jumps to other program locations are easily implemented. Loading a value directly into this PCL register will cause a jump to the specified Program Memory location. However, as the register is only 8-bit wide, only jumps within the current Program Memory page are permitted. When such operations are used, note that a dummy cycle will be inserted.

Look-up Table Registers – TBLP, TBLH

These three special function registers are used to control operation of the look-up table which is stored in the Program Memory. TBLP is the table pointer and indicates the location where the table data is located. The value must be setup before any table read commands are executed. Their value can be changed, for example using the "INC" or "DEC" instructions, allowing for easy table data pointing and reading. TBLH is the location where the high order byte of the table data is stored after a table read data instruction has been executed. Note that the lower order table data byte is transferred to a user defined location.

Status Register – STATUS

This 8-bit register contains the zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), power down flag (PDF), and watchdog time-out flag (TO). These arithmetic/logical operation and system management flags are used to record the status and operation of the microcontroller.

With the exception of the TO and PDF flags, bits in the status register can be altered by instructions like most other registers. Any data written into the status register will not change the TO or PDF flag. In addition, operations related to the status register may give different results due to the different instruction operations. The TO flag can be affected only by a system power-up, a WDT time-out or by executing the "CLR WDT" or "HALT" instruction. The PDF flag is affected only by executing the "HALT" or "CLR WDT" instruction or during a system power-up.

The Z, OV, AC and C flags generally reflect the status of the latest operations.

In addition, on entering an interrupt sequence or executing a subroutine call, the status register will not be pushed onto the stack automatically. If the contents of the status registers are important and if the interrupt routine can change the status register, precautions must be taken to correctly save it. Note that bits 0~3 of the STATUS register are both readable and writeable bits.

STATUS Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	TO	PDF	OV	Z	AC	C
R/W	—	—	R	R	R/W	R/W	R/W	R/W
POR	—	—	0	0	x	x	x	x

"x" unknown

Bit 7 ~ 6 Unimplemented, read as "0"

Bit 5 **TO**: Watchdog Time-Out flag
0: After power up or executing the "CLR WDT" or "HALT" instruction
1: A watchdog time-out occurred.

Bit 4 **PDF**: Power down flag
0: After power up or executing the "CLR WDT" instruction
1: By executing the "HALT" instruction

Bit 3 **OV**: Overflow flag
0: no overflow
1: an operation results in a carry into the highest-order bit but not a carry out of the highest-order bit or vice versa.

Bit 2 **Z**: Zero flag
0: The result of an arithmetic or logical operation is not zero
1: The result of an arithmetic or logical operation is zero

Bit 1 **AC**: Auxiliary flag
0: no auxiliary carry
1: an operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction

Bit 0 **C**: Carry flag
0: no carry-out
1: an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation

C is also affected by a rotate through carry instruction.

EEPROM Data Memory

The HT45F23A contains an area of internal EEPROM Data Memory EEPROM, which stands for Electrically Erasable Programmable Read Only Memory, is by its nature a non-volatile form of re-programmable memory, with data retention even when its power supply is removed. By incorporating this kind of data memory, a whole new host of application possibilities are made available to the designer. The availability of EEPROM storage allows information such as product identification numbers, calibration values, specific user data, system setup data or other product information to be stored directly within the product microcontroller. The process of reading and writing data to the EEPROM memory has been reduced to a very trivial affair.

EEPROM Data Memory Structure

The EEPROM Data Memory capacity is 64×8 bits. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable in the same way as the other types of memory. Read and Write operations to the EEPROM are carried out in single byte operations using an address and data register in Bank 0 and a single control register in Bank 1.

EEPROM Registers

Three registers control the overall operation of the internal EEPROM Data Memory. These are the address register, EEA, the data register, EED and a single control register, EEC. As both the EEA and EED registers are located in Bank 0, they can be directly accessed in the same was as any other Special Function Register. The EEC register however, being located in Bank1, cannot be directly addressed directly and can only be read from or written to indirectly using the MP1 Memory Pointer and Indirect Addressing Register, IAR1. Because the EEC control register is located at address 40H in Bank 1, the MP1 Memory Pointer must first be set to the value 40H and the Bank Pointer register, BP, set to the value, 01H, before any operations on the EEC register are executed.

EEPROM Register List

Name	Bit							
	7	6	5	4	3	2	1	0
EEA	—	—	D5	D4	D3	D2	D1	D0
EED	D7	D6	D5	D4	D3	D2	D1	D0
EEC	—	—	—	—	WREN	WR	RDEN	RD

EEA Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	D5	D4	D3	D2	D1	D0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	x	x	x	x	x	x

"x" unknown

Bit 7 ~ 6 Unimplemented, read as "0"

Bit 5 ~ 0 Data EEPROM address
Data EEPROM address bit 5 ~ bit 0

EEC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	WREN	WR	RDEN	RD
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7 ~ 4 Unimplemented, read as "0"

Bit 3 **WREN**: Data EEPROM Write Enable
0: Disable
1: Enable
This is the Data EEPROM Write Enable Bit which must be set high before Data EEPROM write operations are carried out. Clearing this bit to zero will inhibit Data EEPROM write operations.

Bit 2 **WR**: EEPROM Write Control
0: Write cycle has finished
1: Activate a write cycle
This is the Data EEPROM Write Control Bit and when set high by the application program will activate a write cycle. This bit will be automatically reset to zero by the hardware after the write cycle has finished. Setting this bit high will have no effect if the WREN has not first been set high.

Bit 1 **RDEN**: Data EEPROM Read Enable
0: Disable
1: Enable
This is the Data EEPROM Read Enable Bit which must be set high before Data EEPROM read operations are carried out. Clearing this bit to zero will inhibit Data EEPROM read operations.

Bit 0 **RD**: EEPROM Read Control
0: Read cycle has finished
1: Activate a read cycle
This is the Data EEPROM Read Control Bit and when set high by the application program will activate a read cycle. This bit will be automatically reset to zero by the hardware after the read cycle has finished. Setting this bit high will have no effect if the RDEN has not first been set high.

Note: The WREN, WR, RDEN and RD can not be set to "1" at the same time in one instruction.
The WR and RD can not be set to "1" at the same time.

Reading Data from the EEPROM

To read data from the EEPROM, the read enable bit, RDEN, in the EEC register must first be set high to enable the read function. The EEPROM address of the data to be read must then be placed in the EEA register. If the RD bit in the EEC register is now set high, a read cycle will be initiated. Setting the RD bit high will not initiate a read operation if the RDEN bit has not been set. When the read cycle terminates, the RD bit will be automatically cleared to zero, after which the data can be read from the EED register. The data will remain in the EED register until another read or write operation is executed. The application program can poll the RD bit to determine when the data is valid for reading.

Writing Data to the EEPROM

To write data to the EEPROM, the write enable bit, WREN, in the EEC register must first be set high to enable the write function. The EEPROM address of the data to be written must then be placed in the EEA register and the data placed in the EED register. If the WR bit in the EEC register is now set high, an internal write cycle will then be initiated. Setting the WR bit high will not initiate a write cycle if the WREN bit has not been set. As the EEPROM write cycle is controlled using an internal timer whose operation is asynchronous to microcontroller system clock, a certain time will elapse before the data will have been written into the EEPROM. Detecting when the write cycle has finished can be implemented either by polling the WR bit in the EEC register or by using the EEPROM interrupt. When the write cycle terminates, the WR bit will be automatically cleared to zero by the microcontroller, informing the user that the data has been written to the EEPROM. The application program can therefore poll the WR bit to determine when the write cycle has ended.

Write Protection

Protection against inadvertent write operation is provided in several ways. After the device is powered-on the Write Enable bit in the control register will be cleared preventing any write operations. Also at power-on the Bank Pointer, BP, will be reset to zero, which means that Data Memory Bank 0 will be selected. As the EEPROM control register is located in Bank 1, this adds a further measure of protection against spurious write operations. During normal program operation, ensuring that the Write Enable bit in the control register is cleared will safeguard against incorrect write operations.

EEPROM Interrupt

The EEPROM write or read interrupt is generated when an EEPROM write or read cycle has ended. The EEPROM interrupt must first be enabled by setting the EE2I bit in the relevant interrupt register. However as the EEPROM is contained within a Multi-function Interrupt, the associated multi-function interrupt enable bit must also be set. When an EEPROM write cycle ends, the E2F request flag and its associated multi-function interrupt request flag will both be set. If the global, EEPROM and Multi-function interrupts are enabled and the stack is not full, a jump to the associated Multi-function Interrupt vector will take place. When the interrupt is serviced only the Multi-function interrupt flag will be automatically reset, the EEPROM interrupt flag must be manually reset by the application program. More details can be obtained in the Interrupt section.

Programming Considerations

Care must be taken that data is not inadvertently written to the EEPROM. Protection can be enhanced by ensuring that the Write Enable bit is normally cleared to zero when not writing. Also the Bank Pointer could be normally cleared to zero as this would inhibit access to Bank 1 where the EEPROM control register exist. Although certainly not necessary, consideration might be given in the application program to the checking of the validity of new write data by a simple read back process. The WR bit in the EEC register should be set immediately after the WREN bit is set, otherwise the EEPROM write cycle will not be executed.

Programming Examples

Reading data from the EEPROM – polling method

```
MOV A, EEPROM_ADRES      ; user defined address
MOV EEA, A
MOV A, 040H              ; setup memory pointer MP1
MOV MP1, A               ; MP1 points to EEC register
MOV A, 01H               ; setup Bank Pointer
MOV BP, A
SET IAR1.1               ; set RDEN bit, enable read operations
SET IAR1.0               ; start Read Cycle - set RD bit
BACK:
SZ  IAR1.0               ; check for read cycle end
JMP BACK
CLR IAR1                 ; disable EEPROM read/write
CLR BP
MOV A, EED               ; move read data to register
MOV READ_DATA, A
```

Writing Data to the EEPROM – polling method

```
MOV A, EEPROM_ADRES      ; user defined address
MOV EEA, A
MOV A, EEPROM_DATA       ; user defined data
MOV EED, A
MOV A, 040H              ; setup memory pointer MP1
MOV MP1, A               ; MP1 points to EEC register
MOV A, 01H               ; setup Bank Pointer
MOV BP, A
SET IAR1.3               ; set WREN bit, enable write operations
SET IAR1.2               ; start Write Cycle - set WR bit
BACK:
SZ  IAR1.2               ; check for write cycle end
JMP BACK
CLR IAR1                 ; disable EEPROM read/write
CLR BP
```

Oscillator

Various oscillator options offer the user a wide range of functions according to their various application requirements. The flexible features of the oscillator functions ensure that the best optimisation can be achieved in terms of speed and power saving. Oscillator selections and operation are selected through a combination of configuration options and registers.

Oscillator Overview

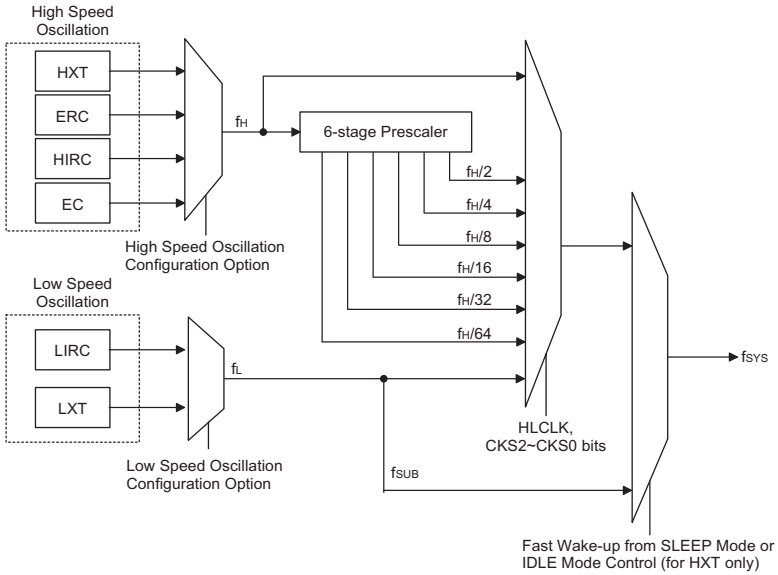
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. External oscillators requiring some external components as well as fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. All oscillator options are selected through the configuration options. The higher frequency oscillators provide higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency oscillators. With the capability of dynamically switching between fast and slow system clock, the device has the flexibility to optimize the performance/ power ratio, a feature especially important in power sensitive portable applications.

Type	Name	Freq.	Pins
External Crystal	HXT	400kHz~8MHz	OSC1/OSC2
External RC	ERC	4MHz	OSC1
Internal High Speed RC	HIRC	910kHz, 2/4/8MHz	—
External Clock	EC	400kHz~8MHz	OSC1
External Low Speed Crystal	LXT	32.768kHz	XT1/XT2
Internal Low Speed RC	LIRC	32kHz	—

Oscillator Types

System Clock Configurations

There are six methods of generating the system clock, four high speed oscillators and two low speed oscillators. The high speed oscillators are the external crystal/ ceramic oscillator, external RC network oscillator, external clock and the internal 910kHz, 2MHz, 4MHz or 8MHz RC oscillator. The two low speed oscillators are the internal 32kHz RC oscillator and the external 32.768kHz crystal oscillator.

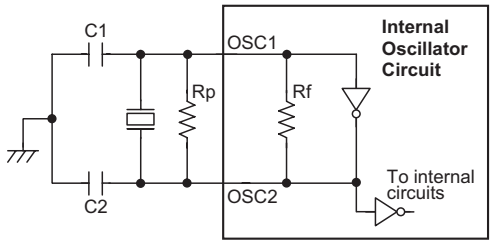


System Clock Configuration

Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the HLCLK bit and CKS2~CKS0 bits in the SMOD register and as the system clock can be dynamically selected. The actual source clock used for each of the high speed and low speed oscillators is chosen via configuration options. The frequency of the slow speed or high speed system clock is also determined using the HLCLK bit and CKS2~CKS0 bits in the SMOD register. Note that two oscillator selections must be made namely one high speed and one low speed system oscillators. It is not possible to choose a no-oscillator selection for either the high or low speed oscillator.

External Crystal/Ceramic Oscillator – HXT

The External Crystal/ Ceramic System Oscillator is one of the high frequency oscillator choices, which is selected via configuration option. For most crystal oscillator configurations, the simple connection of a crystal across OSC1 and OSC2 will create the necessary phase shift and feedback for oscillation, without requiring external capacitors. However, for some crystal types and frequencies, to ensure oscillation, it may be necessary to add two small value capacitors, C1 and C2. Using a ceramic resonator will usually require two small value capacitors, C1 and C2, to be connected as shown for oscillation to occur. The values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification.



Note: 1. Rp is normally not required. C1 and C2 are required.
2. Although not shown OSC1/OSC2 pins have a parasitic capacitance of around 7pF.

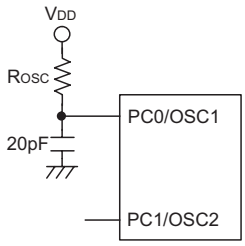
Crystal/Resonator Oscillator – HXT

Crystal Oscillator C1 and C2 Values		
Crystal Frequency	**C1**	**C2**
8MHz	0pF	0pF
4MHz	0pF	0pF
1MHz	100pF	100pF
Note:C1 and C2 values are for guidance only.		

Crystal Recommended Capacitor Values

External RC Oscillator – ERC

Using the ERC oscillator only requires that a resistor, with a value between 56kΩ and 2.4MΩ, is connected between OSC1 and VDD, and a capacitor is connected between OSC1 and ground, providing a low cost oscillator configuration. It is only the external resistor that determines the oscillation frequency; the external capacitor has no influence over the frequency and is connected for stability purposes only. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. As a resistance/frequency reference point, it can be noted that with an external 150kΩ resistor connected and with a 5V voltage power supply and temperature of 25°C degrees, the oscillator will have a frequency of 4MHz within a tolerance of 2%. Here only the OSC1 pin is used, which is shared with I/O pin PC0, leaving pin PC1 free for use as a normal I/O pin.



External RC Oscillator – ERC

External Oscillator – EC

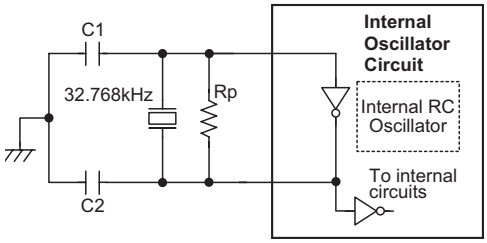
The system clock can also be supplied by an externally supplied clock giving users a method of synchronising their external hardware to the microcontroller operation. This is selected using a configuration option and supplying the clock on pin OSC1. Pin OSC2 should be left floating if the external oscillator is used. The internal oscillator circuit contains a filter circuit to reduce the possibility of erratic operation due to noise on the oscillator pin, however as the filter circuit consumes a certain amount of power, a configuration option exists to turn this filter off. Not using the internal filter should be considered in power sensitive applications and where the externally supplied clock is of a high integrity and supplied by a low impedance source.

Internal RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has four fixed frequencies of either 910kHz, 2MHz, 4MHz or 8MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. As a result, at a power supply of either 3V or 5V and at a temperature of 25°C degrees, the fixed oscillation frequency of 910kHz, 2MHz, 4MHz or 8MHz will have a tolerance within 2%. Note that if this internal system clock option is selected, as it requires no external pins for its operation, I/O pins PC0 and PC1 are free for use as normal I/O pins.

External 32.768kHz Crystal Oscillator – LXT

The External 32.768kHz Crystal System Oscillator is one of the low frequency oscillator choices, which is selected via configuration option. This clock source has a fixed frequency of 32.768kHz and requires a 32.768kHz crystal to be connected between pins XT1 and XT2. The external resistor and capacitor components connected to the 32.768kHz crystal are necessary to provide oscillation. For applications where precise frequencies are essential, these components may be required to provide frequency compensation due to different crystal manufacturing tolerances. During power-up there is a time delay associated with the LXT oscillator waiting for it to start-up.



Note: 1. Rp, C1 and C2 are required.
2. Although not shown pins have a parasitic capacitance of around 7pF.

External LXT Oscillator

When the microcontroller enters the SLEEP or IDLE Mode, the system clock is switched off to stop microcontroller activity and to conserve power. However, in many microcontroller applications it may be necessary to keep the internal timers operational even when the microcontroller is in the SLEEP or IDLE Mode. To do this, another clock, independent of the system clock, must be provided.

However, for some crystals, to ensure oscillation and accurate frequency generation, it is necessary to add two small value external capacitors, C1 and C2. The exact values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer□s specification. The external parallel feedback resistor, Rp, is required.

Some configuration options determine if the XT1/XT2 pins are used for the LXT oscillator or as I/O pins.

- If the LXT oscillator is not used for any clock source, the XT1/XT2 pins can be used as normal I/O pins.
- If the LXT oscillator is used for any clock source, the 32.768kHz crystal should be connected to the XT1/XT2 pins.

LXT Oscillator C1 and C2 Values		
Crystal Frequency	**C1**	**C2**
32.768kHz	10pF	10pF
Note:1. C1 and C2 values are for guidance only. 2. R_P=5M~10MΩ is recommended.		

32.768kHz Crystal Recommended Capacitor Values

LXT Oscillator Low Power Function

The LXT oscillator can function in one of two modes, the Quick Start Mode and the Low Power Mode. The mode selection is executed using the LXTLP bit in the TBC register.

LXTLP Bit	LXT Mode
0	Quick Start
1	Low-power

After power on the LXTLP bit will be automatically cleared to zero ensuring that the LXT oscillator is in the Quick Start operating mode. In the Quick Start Mode the LXT oscillator will power up and stabilise quickly. However, after the LXT oscillator has fully powered up it can be placed into the Low-power mode by setting the LXTLP bit high. The oscillator will continue to run but with reduced current consumption, as the higher current consumption is only required during the LXT oscillator start-up. In power sensitive applications, such as battery applications, where power consumption must be kept to a minimum, it is therefore recommended that the application program sets the LXTLP bit high about 2 seconds after power-on.

It should be noted that, no matter what condition the LXTLP bit is set to, the LXT oscillator will always function normally, the only difference is that it will take more time to start up if in the Low-power mode.

Internal 32kHz Oscillator – LIRC

The Internal 32kHz System Oscillator is one of the low frequency oscillator choices, which is selected via configuration option. It is a fully integrated RC oscillator with a typical frequency of 32kHz at 5V, requiring no external components for its implementation. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. As a result, at a power supply of 5V and at a temperature of 25°C degrees, the fixed oscillation frequency of 32kHz will have a tolerance within 10%.

Supplementary Oscillators

The low speed oscillators, in addition to providing a system clock source are also used to provide a clock source to two other device functions. These are the Watchdog Timer and the Time Base Interrupts.

Operating Modes and System Clocks

Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their nature increase current consumption and of course vice-versa, lower speed clocks reduce current consumption. As Holtek has provided the device with both high and low speed clock sources and the means to switch between them dynamically, the user can optimise the operation of their microcontroller to achieve the best performance/power ratio.

System Clocks

The device has many different clock sources for both the CPU and peripheral function operation. By providing the user with a wide range of clock options using configuration options and register programming, a clock system can be configured to obtain maximum application performance.

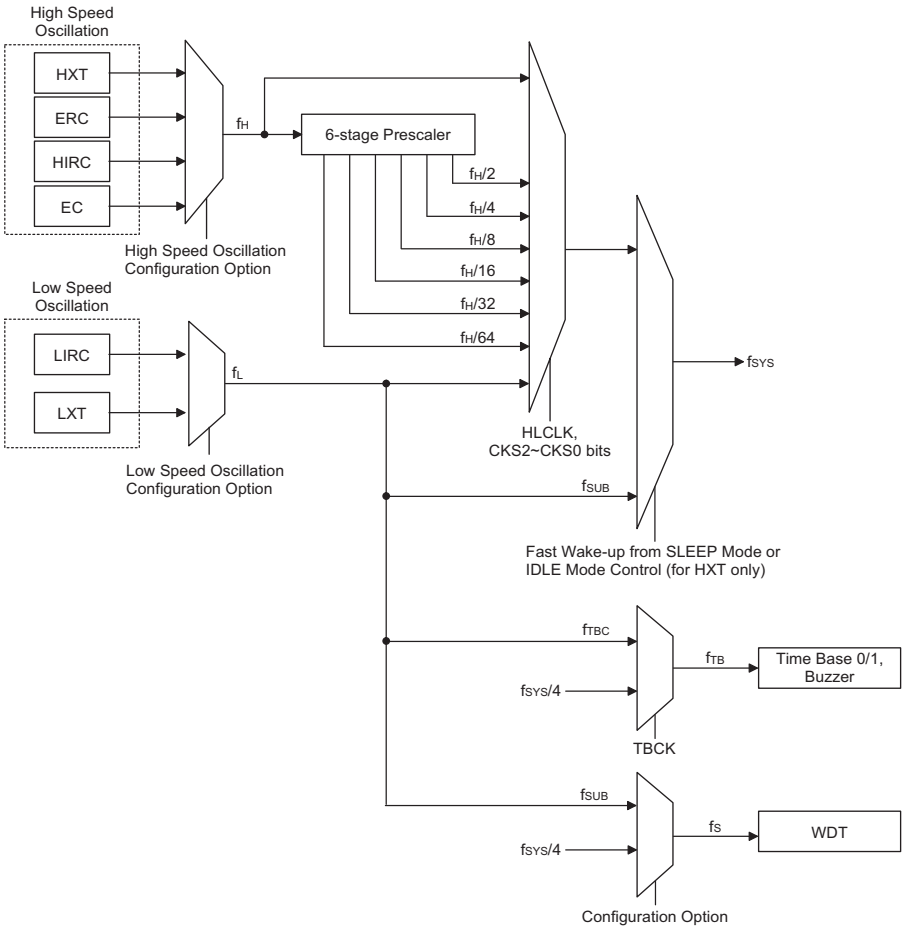
The main system clock, can come from either a high frequency, f_H, or low frequency, f_L, source, and is selected using the HLCLK bit and CKS2~CKS0 bits in the SMOD register. The high speed system clock can be sourced from either an HXT, ERC, EC or HIRC oscillator, selected via a configuration option. The low speed system clock source can be sourced from internal clock f_L. If f_L is selected then it can be sourced by either the LXT or LIRC oscillators, selected via a configuration option. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2$~$f_H/64$.

There are two additional internal clocks for the peripheral circuits, the substitute clock, f_{SUB}, and the Period Time Clock, f_{TBC}. Each of these internal clocks are sourced by either the LXT or LIRC oscillators, selected via configuration options. The f_{SUB} clock is used to provide a substitute clock for the microcontroller just after a wake-up has occurred to enable faster wake-up times.

Together with $f_{SYS}/4$ it is also used as one of the clock sources for the Watchdog timer. The f_{TB} clock is used as a source for the Time Base 0/1 interrupt functions.

System Operation Modes

There are six different modes of operation for the microcontroller, each one with its own special characteristics and which can be chosen according to the specific performance and power requirements of the application. There are two modes allowing normal operation of the microcontroller, the NORMAL Mode and SLOW Mode. The remaining four modes, the SLEEP0, SLEEP1, IDLE0 and IDLE1 Mode are used when the microcontroller CPU is switched off to conserve power.



System Clock Configurations

Note: When the system clock source f_{SYS} is switched to f_L from f_H, the high speed oscillation will stop to conserve the power. Thus there is no f_H~$f_H/64$ for peripheral circuit to use.

Operation Mode	Description				
	CPU	f_{SYS}	f_{SUB}	f_S	f_{TBC}
NORMAL Mode	On	f_H~ $f_H/64$	On	On	On
SLOW Mode	On	f_L	On	On	On
IDLE0 Mode	Off	Off	On	On/Off	On
IDLE1 Mode	Off	On	On	On	On
SLEEP0 Mode	Off	Off	Off	Off	Off
SLEEP1 Mode	Off	Off	On	On	Off

NORMAL Mode

As the name suggests this is one of the main operating modes where the microcontroller has all of its functions operational and where the system clock is provided by one of the high speed oscillators. This mode operates allowing the microcontroller to operate normally with a clock source will come from one of the high speed oscillators, either the HXT, ERC, HIRC or EC oscillators. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 and HLCLK bits in the SMOD register. Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current.

SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from one of the low speed oscillators, either the LXT or the LIRC. Running the microcontroller in this mode allows it to run with much lower operating currents. In the SLOW Mode, the f_H is off.

SLEEP0 Mode

The SLEEP Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SMOD register is low. In the SLEEP0 mode the CPU will be stopped, and the f_{SUB} and f_S clocks will be stopped too, and the Watchdog Timer function is disabled. In this mode, the LVDEN is must set to "0". If the LVDEN is set to "1", it won't enter the SLEEP0 Mode.

SLEEP1 Mode

The SLEEP Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SMOD register is low. In the SLEEP1 mode the CPU will be stopped. However the f_{SUB} and f_S clocks will continue to operate if the LVDEN is "1" or the Watchdog Timer function is enabled and if its clock source is chosen via configuration option to come from the f_{SUB}.

IDLE0 Mode

The IDLE0 Mode is entered when a HALT instruction is executed and when the IDLEN bit in the SMOD register is high and the FSYSON bit in the WDTC register is low. In the IDLE0 Mode the system oscillator will be inhibited from driving the CPU but some peripheral functions will remain operational such as the Watchdog Timer, Time Base 0 and SIM. In the IDLE0 Mode, the system oscillator will be stopped. In the IDLE0 Mode the Watchdog Timer clock, f_S, will either be on or off depending upon the f_S clock source. If the source is $f_{SYS}/4$ then the f_S clock will be off, and if the source comes from f_{SUB} then f_S will be on.

IDLE1 Mode

The IDLE1 Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SMOD register is high and the FSYSON bit in the WDTC register is high. In the IDLE1 Mode the system oscillator will be inhibited from driving the CPU but may continue to provide a clock source to keep some peripheral functions operational such as the Watchdog Timer, Time Base 0 and SIM. In the IDLE1 Mode, the system oscillator will continue to run, and this system oscillator may be high speed or low speed system oscillator. In the IDLE1 Mode the Watchdog Timer clock, f_S, will be on. If the source is $f_{SYS}/4$ then the f_S clock will be on, and if the source comes from f_{SUB} then f_S will be on.

Control Register

A single register, SMOD, is used for overall control of the internal clocks within the device.

SMOD Register

Bit	7	6	5	4	3	2	1	0
Name	CKS2	CKS1	CKS0	FSTEN	LTO	HTO	IDLEN	HLCLK
R/W	R/W	R/W	R/W	R/W	R	R	R/W	R/W
POR	0	0	0	0	0	0	1	1

Bit 7~5 **CKS2~CKS0**: The system clock selection when HLCLK is "0"
000: f_L (f_{LXT} or f_{LIRC})
001: f_L (f_{LXT} or f_{LIRC})
010: $f_H/64$
011: $f_H/32$
100: $f_H/16$

101: $f_H/8$
110: $f_H/4$
111: $f_H/2$

These three bits are used to select which clock is used as the system clock source. In addition to the system clock source, which can be either the LXT or LIRC, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4 **FSTEN**: Fast Wake-up Control (only for HXT)
0: Disable
1: Enable

This is the Fast Wake-up Control bit which determines if the f_{SUB} clock source is initially used after the device wakes up. When the bit is high, the f_{SUB} clock source can be used as a temporary system clock to provide a faster wake up time as the f_{SUB} clock is available.

Bit 3 **LTO**: Low speed system oscillator ready flag
0: Not ready
1: Ready

This is the low speed system oscillator ready flag which indicates when the low speed system oscillator is stable after power on reset or a wake-up has occurred. The flag will be low when in the SLEEP0 Mode but after a wake-up has occurred, the flag will change to a high level after 1024 clock cycles if the LXT oscillator is used and 1~2 clock cycles if the LIRC oscillator is used.

Bit 2 **HTO**: High speed system oscillator ready flag
0: Not ready
1: Ready

This is the high speed system oscillator ready flag which indicates when the high speed system oscillator is stable. This flag is cleared to "0" by hardware when the device is powered on and then changes to a high level after the high speed system oscillator is stable. Therefore this flag will always be read as "1" by the application program after device power-on. The flag will be low when in the SLEEP or IDLE0 Mode but after a wake-up has occurred, the flag will change to a high level after 1024 clock cycles if the HXT oscillator is used and after 15~16 clock cycles if the ERC or HIRC oscillator is used.

Bit 1 **IDLEN**: IDLE Mode control
0: Disable
1: Enable

This is the IDLE Mode Control bit and determines what happens when the HALT instruction is executed. If this bit is high, when a HALT instruction is executed the device will enter the IDLE Mode. In the IDLE1 Mode the CPU will stop running but the system clock will continue to keep the peripheral functions operational, if FSYSON bit is high. If FSYSON bit is low, the CPU and the system clock will all stop in IDLE0 mode. If the bit is low the device will enter the SLEEP Mode when a HALT instruction is executed.

Bit 0 **HLCLK**: system clock selection
0: $f_H/2 \sim f_H/64$ or f_L
1: f_H

This bit is used to select if the f_H clock or the $f_H/2 \sim f_H/64$ or f_L clock is used as the system clock. When the bit is high the f_H clock will be selected and if low the $f_H/2 \sim f_H/64$ or f_L clock will be selected. When system clock switches from the f_H clock to the f_L clock and the f_H clock will be automatically switched off to conserve power.

Fast Wake-up

To minimise power consumption the device can enter the SLEEP or IDLE0 Mode, where the system clock source to the device will be stopped. However when the device is woken up again, it can take a considerable time for the original system oscillator to restart, stabilise and allow normal operation to resume. To ensure the device is up and running as fast as possible a Fast Wake-up function is provided, which allows f_{SUB}, namely either the LXT or LIRC oscillator, to act as a temporary clock to first drive the system until the original system oscillator has stabilised. As the clock source for the Fast Wake-up function is f_{SUB}, the Fast Wake-up function is only available in the SLEEP1 and IDLE0 modes. When the device is woken up from the SLEEP0 mode, the Fast Wake-up function has no effect because the f_{SUB} clock is stopped. The Fast Wake-up enable/disable function is controlled using the FSTEN bit in the SMOD register.

If the HXT oscillator is selected as the NORMAL Mode system clock, and if the Fast Wake-up function is enabled, then it will take one to two t_{SUB} clock cycles of the LIRC or LXT oscillator for the system to wake-up. The system will then initially run under the f_{SUB} clock source until 1024 HXT clock cycles have elapsed, at which point the HTO flag will switch high and the system will switch over to operating from the HXT oscillator.

If the ERC, EC or HIRC oscillators or LIRC oscillator is used as the system oscillator then it will take 15~16 clock cycles of the ERC, EC or HIRC or 1~2 cycles of the LIRC to wake up the system from the SLEEP or IDLE0 Mode. The Fast Wake-up bit, FSTEN will have no effect in these cases.

System Oscillator	FSTEN Bit	Wake-up Time (SLEEP0 Mode)	Wake-up Time (SLEEP1 Mode)	Wake-up Time (IDLE0 Mode)	Wake-up Time (IDLE1 Mode)
HXT	0	1024 HXT cycles	1024 HXT cycles		1~2 HXT cycles
	1	1024 HXT cycles	1~2 f_{SUB} cycles (System runs with f_{SUB} first for 1024 HXT cycles and then switches over to run with the HXT clock)		1~2 HXT cycles
ERC	X	15~16 ERC cycles	15~16 ERC cycles		1~2 ERC cycles
EC	X	15~16 EC cycles	15~16 EC cycles		1~2 EC cycles
HIRC	X	15~16 HIRC cycles	15~16 HIRC cycles		1~2 HIRC cycles
LIRC	X	1~2 LIRC cycles	1~2 LIRC cycles		1~2 LIRC cycles
LXT	X	1024 LTX cycles	1024 LXT cycles		1~2 LXT cycles

Wake-Up Times

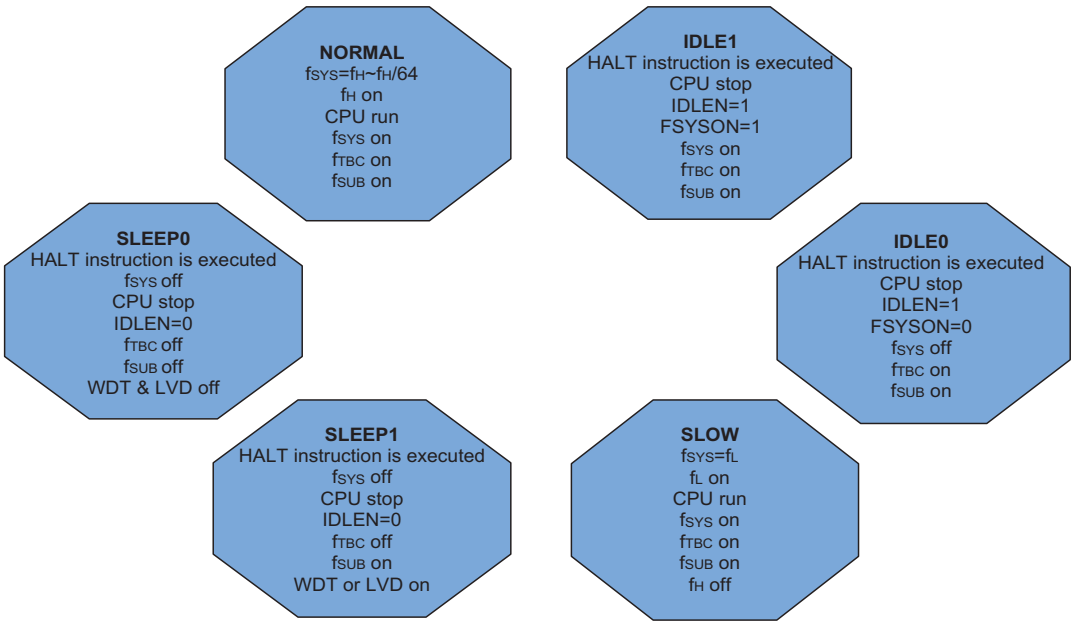
Note that if the Watchdog Timer is disabled, which means that the LXT and LIRC are all both off, then there will be no Fast Wake-up function available when the device wakes-up from the SLEEP0 Mode.

Operating Mode Switching and Wake-up

The device can switch between operating modes dynamically allowing the user to select the best performance/power ratio for the present task in hand. In this way microcontroller operations that do not require high performance can be executed using slower clocks thus requiring less operating current and prolonging battery life in portable applications.

In simple terms, Mode Switching between the NORMAL Mode and SLOW Mode is executed using the HLCLK bit and CKS2~CKS0 bits in the SMOD register while Mode Switching from the NORMAL/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE Mode or the SLEEP Mode is determined by the condition of the IDLEN bit in the SMOD register and FSYSON in the WDTC register.

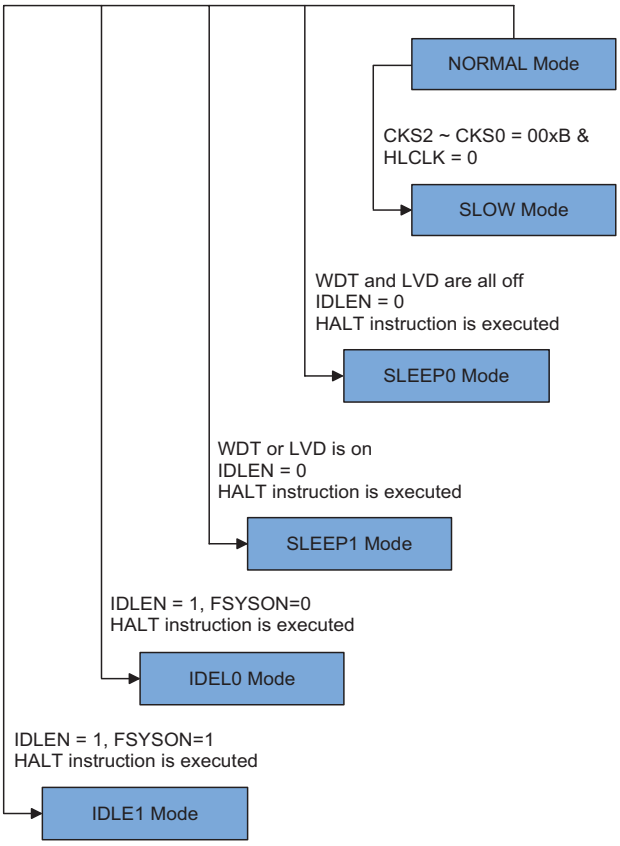
When the HLCLK bit switches to a low level, which implies that clock source is switched from the high speed clock source, f_H, to the clock source, $f_H/2$~$f_H/64$ or f_L. If the clock is from the f_L, the high speed clock source will stop running to conserve power. When this happens it must be noted that the $f_H/16$ and $f_H/64$ internal clock sources will also stop running, which may affect the operation of other internal functions such as the SIM. The accompanying flowchart shows what happens when the device moves between the various operating modes.

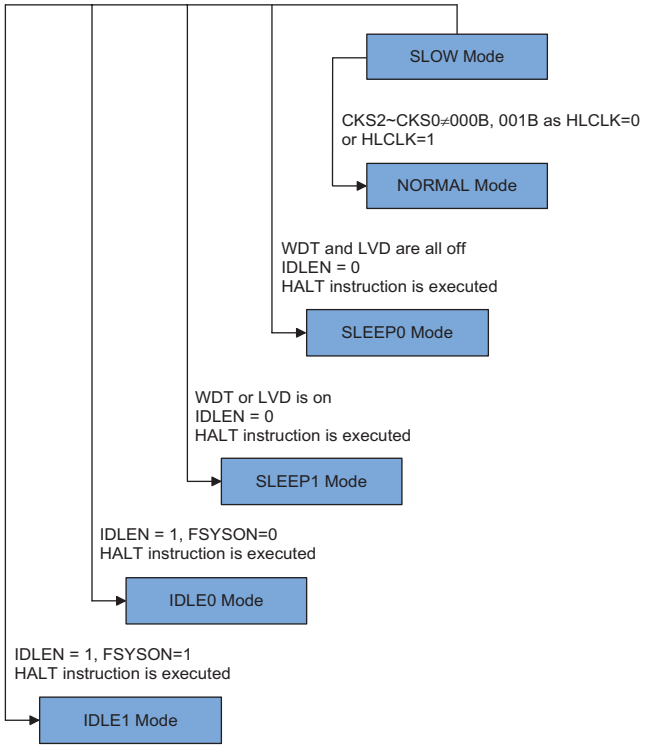


NORMAL Mode to SLOW Mode Switching

When running in the NORMAL Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by set the HLCLK bit to "0" and set the CKS2~CKS0 bits to "000" or "001" in the SMOD register. This will then use the low speed system oscillator which will consume less power. Users may decide to do this for certain operations which do not require high performance and can subsequently reduce power consumption.

The SLOW Mode is sourced from the LXT or the LIRC oscillators and therefore requires these oscillators to be stable before full mode switching occurs. This is monitored using the LTO bit in the SMOD register.


NORMAL Mode
CKS2 ~ CKS0 = 00xB &
HLCLK = 0
SLOW Mode
WDT and LVD are all off
IDLEN = 0
HALT instruction is executed
SLEEP0 Mode
WDT or LVD is on
IDLEN = 0
HALT instruction is executed
SLEEP1 Mode
IDLEN = 1, FSYSON=0
HALT instruction is executed
IDEL0 Mode
IDLEN = 1, FSYSON=1
HALT instruction is executed
IDLE1 Mode


SLOW Mode
CKS2~CKS0≠000B, 001B as HLCLK=0
or HLCLK=1
NORMAL Mode
WDT and LVD are all off
IDLEN = 0
HALT instruction is executed
SLEEP0 Mode
WDT or LVD is on
IDLEN = 0
HALT instruction is executed
SLEEP1 Mode
IDLEN = 1, FSYSON=0
HALT instruction is executed
IDLE0 Mode
IDLEN = 1, FSYSON=1
HALT instruction is executed
IDLE1 Mode

SLOW Mode to NORMAL Mode Switching

In SLOW Mode the system uses either the LXT or LIRC low speed system oscillator. To switch back to the NORMAL Mode, where the high speed system oscillator is used, the HLCLK bit should be set to "1" or HLCLK bit is "0", but CKS2~CKS0 is set to "010", "011", "100", "101", "110" or "111". As a certain amount of time will be required for the high frequency clock to stabilise, the status of the HTO bit is checked. The amount of time required for high speed system oscillator stabilization depends upon which high speed system oscillator type is used.

Entering the SLEEP0 Mode

There is only one way for the device to enter the SLEEP0 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in SMOD register equal to "0" and the WDT and LVD both off. When this instruction is executed under the conditions described above, the following will occur:

- The system clock, WDT clock and Time Base clock will be stopped and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and stopped no matter if the WDT clock source originates from the f_{SUB} clock or from the system clock.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Entering the SLEEP1 Mode

There is only one way for the device to enter the SLEEP1 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in SMOD register equal to "0" and the WDT or LVD on. When this instruction is executed under the conditions described above, the following will occur:

- The system clock and Time Base clock will be stopped and the application program will stop at the "HALT" instruction, but the WDT or LVD will remain with the clock source coming from the f_{SUB} clock.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting if the WDT clock source is selected to come from the f_{SUB} clock as the WDT is enabled.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Entering the IDLE0 Mode

There is only one way for the device to enter the IDLE0 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in SMOD register equal to "1" and the FSYSON bit in WDTC register equal to "0". When this instruction is executed under the conditions described above, the following will occur:

- The system clock will be stopped and the application program will stop at the "HALT" instruction, but the Time Base clock and fSUB clock will be on.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting if the WDT clock source is selected to come from the f_{SUB} clock and the WDT is enabled. The WDT will stop if its clock source originates from the system clock.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Entering the IDLE1 Mode

There is only one way for the device to enter the IDLE1 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in SMOD register equal to "1" and the FSYSON bit in WDTC register equal to "1". When this instruction is executed under the with conditions described above, the following will occur:

- The system clock and Time Base clock and f_{SUB} clock will be on and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting if the WDT is enabled regardless of the WDT clock source which originates from the f_{SUB} clock or from the system clock.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Standby Current Considerations

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of the device to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 Mode, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised. Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to device which have different package types, as there may be unbonbed pins. These must either be setup as outputs or if setup as inputs must have pull-high resistors connected.

Care must also be taken with the loads, which are connected to I/O pins, which are setup as outputs. These should be placed in a condition in which minimum current is drawn or connected only to external circuits that do not draw current, such as other CMOS inputs. Also note that additional standby current will also be required if the configuration options have enabled the LXT or LIRC oscillator.

In the IDLE1 Mode the system oscillator is on, if the system oscillator is from the high speed system oscillator, the additional standby current will also be perhaps in the order of several hundred micro-amps

Wake-up

After the system enters the SLEEP or IDLE Mode, it can be woken up from one of various sources listed as follows:

- An external reset
- An external falling edge on Port A
- A system interrupt
- A WDT overflow

If the system is woken up by an external reset, the device will experience a full system reset, however, if the device is woken up by a WDT overflow, a Watchdog Timer reset will be initiated. Although both of these wake-up methods will initiate a reset operation, the actual source of the wake-up can be determined by examining the TO and PDF flags. The PDF flag is cleared by a system power-up or executing the clear Watchdog Timer instructions and is set when executing the "HALT" instruction. The TO flag is set if a WDT time-out occurs, and causes a wake-up that only resets the Program Counter and Stack Pointer, the other flags remain in their original status.

Each pin on Port A can be setup using the PAWU register to permit a negative transition on the pin to wake-up the system. When a Port A pin wake-up occurs, the program will resume execution at the instruction following the "HALT" instruction. If the system is woken up by an interrupt, then two possible situations may occur. The first is where the related interrupt is disabled or the interrupt is enabled but the stack is full, in which case the program will resume execution at the instruction following the "HALT" instruction. In this situation, the interrupt which woke-up the device will not be immediately serviced, but will rather be serviced later when the related interrupt is finally enabled or when a stack level becomes free. The other situation is where the related interrupt is enabled and the stack is not full, in which case the regular interrupt response takes place. If an interrupt request flag is set high before entering the SLEEP or IDLE Mode, the wake-up function of the related interrupt will be disabled.

Programming Considerations

The HXT and LXT oscillators both use the same SST counter. For example, if the system is woken up from the SLEEP0 Mode and both the HXT and LXT oscillators need to start-up from an off state. The LXT oscillator uses the SST counter after HXT oscillator has finished its SST period.

- If the device is woken up from the SLEEP0 Mode to the NORMAL Mode, the high speed system oscillator needs an SST period. The device will execute first instruction after HTO is "1". At this time, the LXT oscillator may not be stability if f_{SUB} is from LXT oscillator. The same situation occurs in the power-on state. The LXT oscillator is not ready yet when the first instruction is executed.
- If the device is woken up from the SLEEP1 Mode to NORMAL Mode, and the system clock source is from HXT oscillator and FSTEN is "1", the system clock can be switched to the LXT or LIRC oscillator after wake up.
- There are peripheral functions, such as WDT, TMs and SIM, for which the f_{SYS} is used. If the system clock source is switched from f_H to f_L, the clock source to the peripheral functions mentioned above will change accordingly.
- The on/off condition of f_{SUB} and f_S depends upon whether the WDT is enabled or disabled as the WDT clock source is selected from f_{SUB}.

Watchdog Timer

The Watchdog Timer is provided to prevent program malfunctions or sequences from jumping to unknown locations, due to certain uncontrollable external events such as electrical noise.

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal clock, f_S, which is in turn supplied by one of two sources selected by configuration option: f_{SUB} or $f_{SYS}/4$. The f_{SUB} clock can be sourced from either the LXT or LIRC oscillators, again chosen via a configuration option. The Watchdog Timer source clock is then subdivided by a ratio of 2^{13} to 2^{20} to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register. The LIRC internal oscillator has an approximate period of 32kHz at a supply voltage of 5V.

However, it should be noted that this specified internal clock period can vary with VDD, temperature and process variations. The LXT oscillator is supplied by an external 32.768kHz crystal. The other Watchdog Timer clock source option is the $f_{SYS/4}$ clock. The Watchdog Timer clock source can originate from its own internal LIRC oscillator, the LXT oscillator or fSYS/4. It is divided by a value of 2^{13} to 2^{20}, using the WS2~WS0 bits in the WDTC register to obtain the required Watchdog Timer time-out period.

Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as the enable/disable operation. This register together with several configuration options control the overall operation of the Watchdog Timer.

WDTC Register

Bit	7	6	5	4	3	2	1	0
Name	FSYSON	WS2	WS1	WS0	WDTEN3	WDTEN2	WDTEN1	WDTEN0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	1	1	1	0	1	0

Bit 7 **FSYSON**: f_{SYS} Control in IDLE Mode
0: Disable
1: Enable

Bit 6 ~ 4 **WS2, WS1, WS0**: WDT time-out period selection
000: $2^{13}/f_S$
001: $2^{14}/f_S$
010: $2^{15}/f_S$
011: $2^{16}/f_S$
100: $2^{17}/f_S$
101: $2^{18}/f_S$
110: $2^{19}/f_S$
111: $2^{20}/f_S$
These three bits determine the division ratio of the Watchdog Timer source clock, which in turn determines the timeout period.

Bit 3 ~ 0 **WDTEN3, WDTEN2, WDTEN1, WDTEN0**: WDT Software Control
1010: Disable
Other: Enable

Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instructions. If the program malfunctions for whatever reason, jumps to an unkown location, or enters an endless loop, these clear instructions will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. Some of the Watchdog Timer options, such as enable/disable, clock source selection and clear instruction type are selected using configuration options. In addition to a configuration option to enable/disable the Watchdog Timer, there are also four bits, WDTEN3~WDTEN0, in the WDTC register to offer an additional enable/disable control of the Watchdog Timer. To disable the Watchdog Timer, as well as the configuration option being set to disable, the WDTEN3~WDTEN0 bits must also be set to a specific value of "1010". Any other values for these bits will keep the Watchdog Timer enabled, irrespective of the configuration enable/disable setting. After power on these bits will have the value of 1010. If the Watchdog Timer is used it is recommended that they are set to a value of 0101 for maximum noise immunity. Note that if the Watchdog Timer has been disabled, then any instruction relating to its operation will result in no operation.

WDT Configuration Option	WDTEN3~WDTEN0 Bits	WDT
WDT Enable	xxxx	Enable
WDT Disable	Except 1010	Enable
WDT Disable	1010	Disable

Watchdog Timer Enable/Disable Control

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. However, if the system is in the SLEEP or IDLE Mode, when a Watchdog Timer time-out occurs, the TO bit in the status register will be set and only the Program Counter and Stack Pointer will be reset. Three methods can be adopted to clear the contents of the Watchdog Timer. The first is an external hardware reset, which means a low level on the RES pin, the second is using the Watchdog Timer software clear instructions and the third is via a HALT instruction.

There are two methods of using software instructions to clear the Watchdog Timer, one of which must be chosen by configuration option. The first option is to use the single "CLR WDT" instruction while the second is to use the two commands "CLR WDT1" and "CLR WDT2". For the first option, a simple execution of "CLR WDT" will clear the WDT while for the second option, both "CLR WDT1" and "CLR WDT2" must both be executed alternately to successfully clear the Watchdog Timer. Note that for this second option, if "CLR WDT1" is used to clear the Watchdog Timer, successive executions of this instruction will have no effect, only the execution of a "CLR WDT2" instruction will clear the Watchdog Timer. Similarly after the "CLR WDT2" instruction has been executed, only a successive "CLR WDT1" instruction can clear the Watchdog Timer.

The maximum time out period is when the 2^{20} division ratio is selected. As an example, with a 32.768kHz LXT oscillator as its source clock, this will give a maximum watchdog period of around 32 seconds for the 2^{20} division ratio, and a minimum timeout of 250ms for the 2^{13} division ration. If the $f_{SYS}/4$ clock is used as the Watchdog Timer clock source, it should be noted that when the system enters the SLEEP or IDLE0 Mode, then the instruction clock is stopped and the Watchdog Timer may lose its protecting purposes. For systems that operate in noisy environments, using the f_{SUB} clock source is strongly recommended.

Reset and Initialisation

A reset function is a fundamental part of any microcontroller ensuring that the device can be set to some predetermined condition irrespective of outside parameters. The most important reset condition is after power is first applied to the microcontroller. In this case, internal circuitry will ensure that the microcontroller, after a short delay, will be in a well defined state and ready to execute the first program instruction. After this power-on reset, certain important internal registers will be set to defined states before the program commences. One of these registers is the Program Counter, which will be reset to zero forcing the microcontroller to begin program execution from the lowest Program Memory address.

In addition to the power-on reset, situations may arise where it is necessary to forcefully apply a reset condition when the microcontroller is running. One example of this is where after power has been applied and the microcontroller is already running, the $\overline{RES}$ line is forcefully pulled low. In such a case, known as a normal operation reset, some of the microcontroller registers remain unchanged allowing the microcontroller to proceed with normal operation after the reset line is allowed to return high.

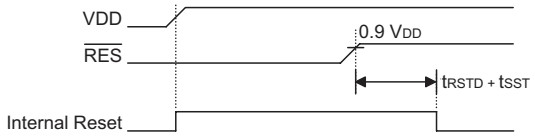
Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result in different register conditions being setup. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, similar to the $\overline{RES}$ reset is implemented in situations where the power supply voltage falls below a certain threshold.

Reset Functions

There are five ways in which a microcontroller reset can occur, through events occurring both internally and externally:

Power-on Reset

The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions. All the I/O port and port control registers will power up in a high condition ensuring that all pins will be first set to inputs.



Note: t_{RSTD} is power-on delay, typical time=100ms

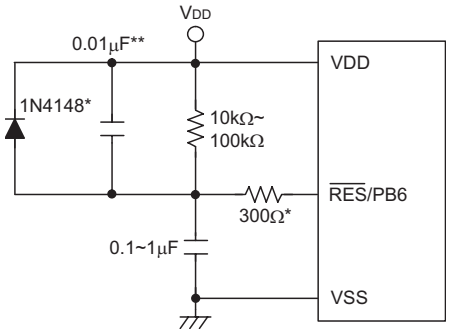
Power-On Reset Timing Chart

$\overline{RES}$ Pin

As the reset pin is shared with PB.6, the reset function must be selected using a configuration option. Although the microcontroller has an internal RC reset function, if the VDD power supply rise time is not fast enough or does not stabilise quickly at power-on, the internal reset function may be incapable of providing proper reset operation. For this reason it is recommended that an external RC network is connected to the $\overline{RES}$ pin, whose additional time delay will ensure that the RES pin remains low for an extended period to allow the power supply to stabilise. During this time delay, normal operation of the microcontroller will be inhibited. After the $\overline{RES}$ line reaches a certain voltage value, the reset delay time t_{RSTD} is invoked to provide an extra delay time after which the microcontroller will begin normal operation. The abbreviation SST in the figures stands for System Start-up Timer.

For most applications a resistor connected between VDD and the $\overline{RES}$ pin and a capacitor connected between VSS and the $\overline{RES}$ pin will provide a suitable external reset circuit. Any wiring connected to the $\overline{RES}$ pin should be kept as short as possible to minimise any stray noise interference.

For applications that operate within an environment where more noise is present the Enhanced Reset Circuit shown is recommended.

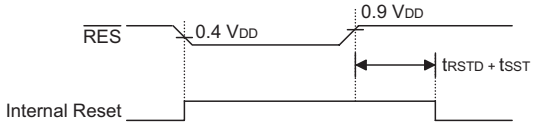


Note: "*" It is recommended that this component is added for added ESD protection.
"**" It is recommended that this component is added in environments where power line noise is significant.

External $\overline{RES}$ Circuit

More information regarding external reset circuits is located in Application Note HA0075E on the Holtek website.

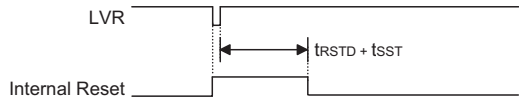
Pulling the $\overline{RES}$ Pin low using external hardware will also execute a device reset. In this case, as in the case of other resets, the Program Counter will reset to zero and program execution initiated from this point.



$\overline{RES}$ Reset Timing Chart

Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device, which is selected via a configuration option. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery, the LVR will automatically reset the device internally. The LVR includes the following specifications: For a valid LVR signal, a low voltage, i.e., a voltage in the range between 0.9V~VLVR must exist for greater than the value t_{LVR} specified in the A.C. characteristics. If the low voltage state does not exceed t_{LVR}, the LVR will ignore it and will not perform a reset function. One of a range of specified voltage values for V_{LVR} can be selected using configuration options.

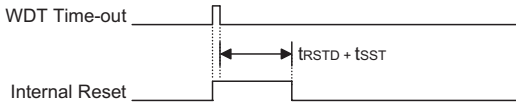


Note: t_{RSTD} is power-on delay, typical time=100ms

Low Voltage Reset Timing Chart

Watchdog Time-out Reset during Normal Operation

The Watchdog time-out Reset during normal operation is the same as a hardware $\overline{RES}$ pin reset except that the Watchdog time-out flag TO will be set to "1".

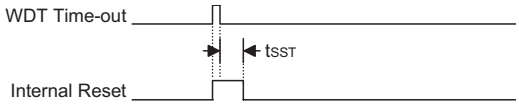


Note: t_{RSTD} is power-on delay, typical time=100ms

WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

The Watchdog time-out Reset during SLEEP or IDLE Mode is a little different from other kinds of reset. Most of the conditions remain unchanged except that the Program Counter and the Stack Pointer will be cleared to "0" and the TO flag will be set to "1". Refer to the A.C. Characteristics for t_{SST} details.



WDT Time-out Reset during Idle/Sleep Timing Chart

Note: The t_{SST} is 15~16 clock cycles if the system clock source is provided by ERC or HIRC.
The t_{SST} is 1024 clock for HXT or LXT.
The t_{SST} is 1~2 clock for LIRC.

Reset Initial Conditions

The different types of reset described affect the reset flags in different ways. These flags, known as PDF and TO are located in the status register and are controlled by various microcontroller operations, such as the SLEEP or IDLE Mode function or Watchdog Timer. The reset flags are shown in the table:

TO	PDF	RESET Conditions
0	0	Power-on reset
u	u	RES or LVR reset during NORMAL or SLOW Mode operation
1	u	WDT time-out reset during NORMAL or SLOW Mode operation
1	1	WDT time-out reset during IDLE or SLEEP Mode operation

Note: "u" stands for unchanged

The following table indicates the way in which the various components of the microcontroller are affected after a power-on reset occurs.

Item	Condition After RESET
Program Counter	Reset to zero
Interrupts	All interrupts will be disabled
WDT	Clear after reset, WDT begins counting
Timer/Event Counter	Timer Counter will be turned off
Input/Output Ports	I/O ports will be setup as inputs
Stack Pointer	Stack Pointer will point to the top of the stack

The different kinds of resets all affect the internal registers of the microcontroller in different ways. To ensure reliable continuation of normal program execution after a reset occurs, it is important to know what condition the microcontroller is in after a particular reset occurs. The following table describes how each type of reset affects each of the microcontroller internal registers. Note that where more than one package type exists the table will reflect the situation for the larger package type.

Register	Power-on Reset	$\overline{\text{RES}}$ or LVR Reset	WDT Time-out (Normal Operation)	WDT Time-out (Idle/Sleep)
PCL	0000 0000	0000 0000	0000 0000	0000 0000
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
BP	- - - - - - - 0	- - - - - - - 0	- - - - - - - 0	- - - - - - - u
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	- xxx xxxx	- uuu uuuu	- uuu uuuu	- uuu uuuu
STATUS	- - 00 xxxx	- - uu uuuu	- - 1u uuuu	- - 11 uuuu
SMOD	0000 0011	0000 0011	0000 0011	uuuu uuuu
LVDC	- - 00 - 000	- - 00 - 000	- - 00 - 000	- -uu - uuu
INTEDGE	- - - - 0000	- - - - 0000	- - - - 0000	- - - - uuuu
WDTC	0111 1010	0111 1010	0111 1010	uuuu uuuu
INTC0	- 000 0000	- 000 0000	- 000 0000	- uuu uuuu
INTC1	- 000 - 000	- 000 - 000	- 000 - 000	- uuu - uuu
MFIC0	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFIC1	- 000 - 000	- 000 - 000	- 000 - 000	- uuu - uuu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	- 111 1111	- 111 1111	- 111 1111	- uuu - uuu
PBC	- 111 1111	- 111 1111	- 111 1111	- uuu - uuu
PC	- 111 1111	- 111 1111	- 111 1111	- uuu - uuu
PCC	- 111 1111	- 111 1111	- 111 1111	- uuu - uuu
PAWU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBPU	- 000 0000	- 000 0000	- 000 0000	- uuu uuuu
PCPU	- 000 0000	- 000 0000	- 000 0000	- uuu uuuu
PWM0	0000 0000	0000 0000	0000 0000	uuuu uuuu
PWM1	0000 0000	0000 0000	0000 0000	uuuu uuuu
MISC	0000 - - 00	0000 - - 00	0000 - - 00	uuuu - - uu
ADPCR	- - 00 0000	- - 00 0000	- - 00 0000	- - uu uuuu
ADRL	xxxx - - - -	xxxx - - - -	xxxx - - - -	uuuu - - - -
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	01 - - -000	01 - - -000	01 - - -000	uuu - - -uu
ACSR	100 - - 000	100 - - 000	100 - - 000	uuu - - uuu
SIMC0	1110 000 -	1110 000 -	1110 000 -	uuuu uuuu
SIMC1	1000 0001	1000 0001	1000 0001	uuuu uuuu
SIMD	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu

Register	Power-on Reset	$\overline{RES}$ or LVR Reset	WDT Time-out (Normal Operation)	WDT Time-out (Idle/Sleep)
SIMA/SIMC2	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR0	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	00- 0 1000	00- 0 1000	00- 0 1000	uu- u uuuu
TMR1L	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1H	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	0000 1- - -	0000 1- - -	0000 1- - -	uuuu u- - -
EEA	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
EED	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
EEC	- - - - 0000	- - - - 0000	- - - - 0000	- - - - uuuu
LCDC	- 000 0000	- 000 0000	- 000 0000	- uuu uuuu
LDOC	- - -0 0000	- - -0 0000	- - -0 0000	- - -u uuuu
DACTRL	000 - - - -0	000 - - - -0	000 - - - -0	uuu - - - -u
DAL	0000 - - - -	0000 - - - -	0000 - - - -	uuuu - - - -
DAH	0000 0000	0000 0000	0000 0000	uuuu uuuu
CMP1C0	0001 0000	0001 0000	0001 0000	uuuu uuuu
CMP1C1	1- - - 0010	1- - - 0010	1- - - 0010	1- - - uuuu
CMP2C0	0001 0000	0001 0000	0001 0000	uuuu uuuu
CMP2C1	00 - - 0010	00 - - 0010	00 - - 0010	uu- - uuuu
OPA1C0	0 - - - - - - -	0 - - - - - - -	0 - - - - - - -	u - - - - - - -
OPA1C1	0000 1100	0000 1100	0000 1100	uuuu uuuu
OPA2C0	0 - - - - - - -	0 - - - - - - -	0 - - - - - - -	u - - - - - - -
OPA2C1	0000 1100	0000 1100	0000 1100	uuuu uuuu
OPA2C2	00 - - 0000	00 - - 0000	00 - - 0000	uu - - uuuu
TBC	0011 0111	0011 0111	0011 0111	uuuu uuuu
BPCTL	0000 0000	0000 0000	0000 0000	uuuu uuuu

Note: "-" not implemented

"u" means "unchanged"

"x" means "unknown"

Input/Output Ports

Holtek microcontrollers offer considerable flexibility on their I/O ports. With the input or output designation of every pin fully under user program control, pull-high selections for all ports and wake-up selections on certain pins, the user is provided with an I/O structure to meet the needs of a wide range of application possibilities.

The device provide bidirectional input/output lines labeled with port names PA~PC. These I/O ports are mapped to the RAM Data Memory with specific addresses as shown in the Special Purpose Data Memory table. All of these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, which means the inputs must be ready at the T2 rising edge of instruction "MOV A,[m]", where m denotes the port address. For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAWU	D7	D6	D5	D4	D3	D2	D1	D0
PAPU	D7	D6	D5	D4	D3	D2	D1	D0
PA	D7	D6	D5	D4	D3	D2	D1	D0
PAC	D7	D6	D5	D4	D3	D2	D1	D0
PBPU	—	D6	D5	D4	D3	D2	D1	D0
PB	—	D6	D5	D4	D3	D2	D1	D0
PBC	—	D6	D5	D4	D3	D2	D1	D0
PCPU	—	D6	D5	D4	D3	D2	D1	D0
PC	—	D6	D5	D4	D3	D2	D1	D0
PCC	—	D6	D5	D4	D3	D2	D1	D0

Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, when configured as an input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selectable via a register known as PAPU, PBPU and PCPU located in the Data Memory. The pull-high resistors are implemented using weak PMOS transistors.

PAPU Register

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **PAPU**: I/O Port bit 7 ~ bit 0 Pull-High Control
0: Disable
1: Enable

PBPU, PCPU Register

Bit	7	6	5	4	3	2	1	0
Name	—	D6	D5	D4	D3	D2	D1	D0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6~0 **PBPU, PCPU**: I/O Port bit 6 ~ bit 0 Pull-High Control
0: Disable
1: Enable

Port A Wake-up

The HALT instruction forces the microcontroller into the SLEEP or IDLE Mode which preserves power, a feature that is important for battery and other low-power applications. Various methods exist to wake-up the microcontroller, one of which is to change the logic condition on one of the Port A pins from high to low. This function is especially suitable for applications that can be woken up via external switches. Each pin on Port A can be selected individually to have this wake-up feature using the PAWU register.

PAWU Register

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **PAWU**: Port A bit 7 ~ bit 0 Wake-up Control
0: Disable
1: Enable

I/O Port Control Registers

Each I/O port has its own control register known as PAC~PCC, to control the input/output configuration. With this control register, each CMOS output or input can be reconfigured dynamically under software control. Each pin of the I/O ports is directly mapped to a bit in its associated port control register. For the I/O pin to function as an input, the corresponding bit of the control register must be written as a a "1". This will then allow the logic state of the input pin to be directly read by instructions. When the corresponding bit of the control register is written as a "0", the I/O pin will be setup as a CMOS output. If the pin is currently setup as an output, instructions can still be used to read the output register.

However, it should be noted that the program will in fact only read the status of the output data latch and not the actual logic status of the output pin.

PAC Register

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

PBC, PCC Register

Bit	7	6	5	4	3	2	1	0
Name	—	D6	D5	D4	D3	D2	D1	D0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	1	1	1	1	1	1	1

Bit 7 Unimplemented, read as "0"

Bit 6~0 **PBC, PCC**: I/O Port bit 6 ~ bit 0 Input/Output Control
0: Output
1: Input

Port B NMOS Open Drain Control Register

Port B pins PB0~PB3 can be setup as open drain structures. This is implemented using the ODE0~ODE3 bits in the MISC register.

MISC Register

Bit	7	6	5	4	3	2	1	0
Name	ODE3	ODE2	ODE1	ODE0	—	—	PFDSEL	PFDEN
R/W	R/W	R/W	R/W	R/W	—	—	R/W	R/W
POR	0	0	0	0	—	—	0	0

Bit 7 **ODE3**: PB3 Open Drain Control
0: disable
1: enable

Bit 6 **ODE2**: PB2 Open Drain Control
0: disable
1: enable

Bit 5 **ODE1**: PB1 Open Drain Control
0: disable
1: enable

Bit 4 **ODE0**: PB0 Open Drain Control
0: disable
1: enable

Bit 3~2 Unimplemented, read as "0"

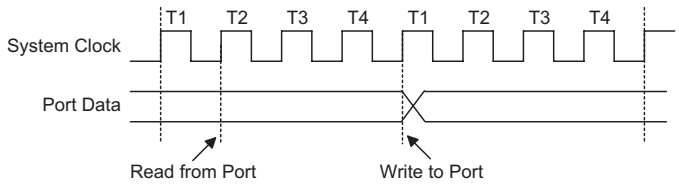
Bit 1~0 **PFDSEL, PFDEN**: PFD related control - described elsewhere

I/O Pin Structures

The accompanying diagram illustrates the internal structures of some generic I/O pin types. As the exact logical construction of the I/O pin will differ from these drawings, they are supplied as a guide only to assist with the functional understanding of the I/O pins. The wide range of pin-shared structures does not permit all types to be shown.

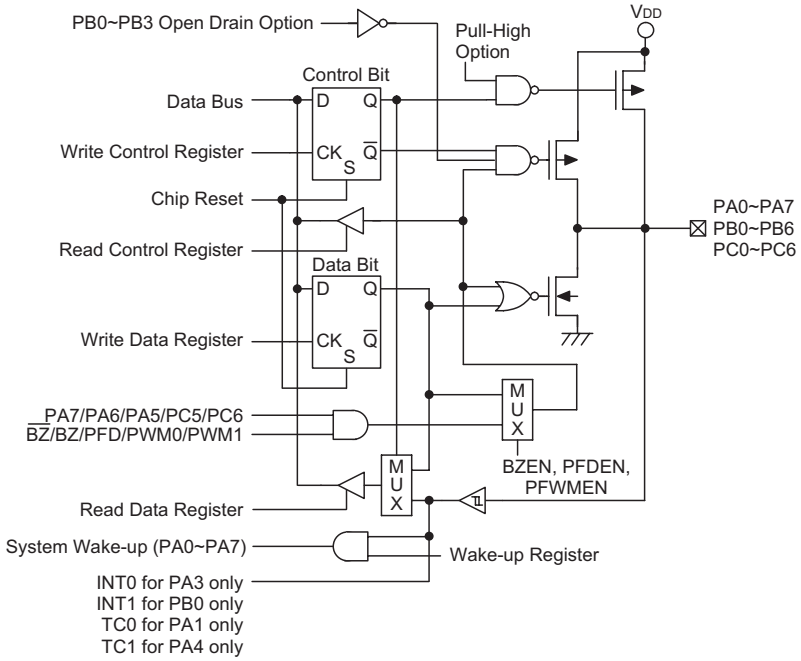
Programming Considerations

Within the user program, one of the first things to consider is port initialisation. After a reset, all of the I/O data and port control registers will be set high. This means that all I/O pins will default to an input state, the level of which depends on the other connected circuitry and whether pull-high selections have been chosen. If the port control registers, PAC~PCC, are then programmed to setup some pins as outputs, these output pins will have an initial high output value unless the associated port data registers, PA~PC, are first programmed. Selecting which pins are inputs and which are outputs can be achieved byte-wide by loading the correct values into the appropriate port control register or by programming individual bits in the port control register using the "SET[m].i" and "CLR[m].i" instructions. Note that when using these bit control instructions, a read-modify-write operation takes place. The microcontroller must first read in the data on the entire port, modify it to the required new bit values and then rewrite this data back to the output ports.



Read Modify Write Timing

Port A has the additional capability of providing wake-up functions. When the device is in the SLEEP or IDLE Mode, various methods are available to wake the device up. One of these is a high to low transition of any of the Port A pins. Single or multiple pins on Port A can be setup to have this function. In addition, the Port B pins also provide Open Drain I/O structure options which can be controlled by the specific register.



Generic Input/Output Ports

Timer/Event Counters

The provision of timers form an important part of any microcontroller, giving the designer a means of carrying out time related functions. The device contain one 8-bit and one 16-bit count-up timer. As each timer has four different operating modes, they can be configured to operate as a general timer, an external event counter, an internal event counter for comparator, or as a pulse width measurement device. The provision of a prescaler to the clock circuitry of the 8-bit Timer/Event Counter also gives added range to this timer.

There are two types of registers related to the Timer/Event Counters. The first are the registers that contain the actual value of the Timer/Event Counter and into which an initial value can be preloaded. Reading from these registers retrieves the contents of the Timer/Event Counter. The second type of associated register is the Timer Control Register which defines the timer options and determines how the Timer/Event Counter is to be used. The Timer/Event Counters can have the their clock configured to come from an internal clock source. In addition, their clock source can also be configured to come from an external timer pin.

Configuring the Timer/Event Counter Input Clock Source

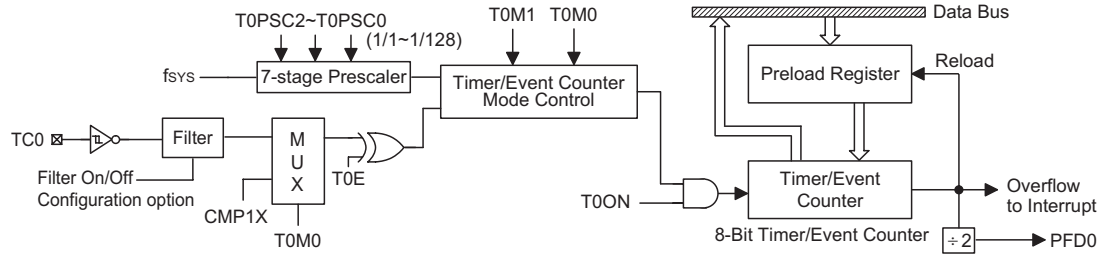
The internal timer's clock can originate from various sources. The system clock source is used when the Timer/Event Counter is in the timer mode or in the pulse width measurement mode. For Timer/Event Counter 0 this internal clock source is f_{SYS} which is also divided by a prescaler, the division ratio of which is conditioned by the Timer Control Register, TMR0C, bits T0PSC0~ T0PSC2. For Timer/Event Counter 1 this internal clock source can be chosen from a combination of internal clocks using a configuration option and the T1S bit in the TMR1C register.

An external clock source is used when the timer is in the event counting mode, the clock source being provided on an external timer pin TC0 or TC1, depending upon which timer is used. Depending upon the condition of the T0E or T1E bit, each high to low, or low to high transition on the external timer pin will increment the counter by one.

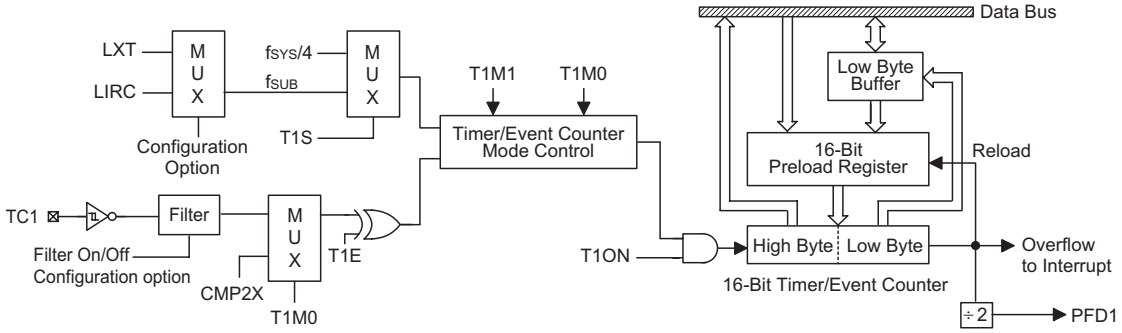
Timer Registers – TMR0, TMR1L, TMR1H

The timer registers are special function registers located in the Special Purpose Data Memory and is the place where the actual timer value is stored. For the 8-bit Timer/Event Counter 0, this register is known as TMR0. For 16-bit Timer/Event Counter 1, the timer registers are known as TMR1L and TMR1H. The value in the timer registers increases by one each time an internal clock pulse is received or an external transition occurs on the external timer pin. The timer will count from the initial value loaded by the preload register to the full count of FFH for the 8-bit timer or FFFFH for the 16-bit timer at which point the timer overflows and an internal interrupt signal is generated. The timer value will then be reset with the initial preload register value and continue counting.

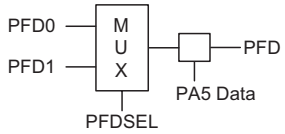
To achieve a maximum full range count of FFH for the 8-bit timer or FFFFH for the 16-bit timer, the preload registers must first be cleared to all zeros. It should be noted that after power-on, the preload register will be in an unknown condition. Note that if the Timer/Event Counter is switched off and data is written to its preload registers, this data will be immediately written into the actual timer registers. However, if the Timer/Event Counter is enabled and counting, any new data written into the preload data registers during this period will remain in the preload registers and will only be written into the timer registers the next time an overflow occurs.



8-bit Timer/Event Counter 0 Structure



16-bit Timer/Event Counter 1 Structure



Note: 1. The PFD clock source, PFD0 or PFD1, which is from Timer0 or Timer1, is selected by PFDSEL bit in MISC register.

2. The output is controlled by PA5 data.

3. CMP1X is comparator 1 output.

4. CMP2X is comparator 2 output.

For the 16-bit Timer/Event Counter which has both low byte and high byte timer registers, accessing these registers is carried out in a specific way. It must be noted when using instructions to preload data into the low byte timer register, namely TMR1L, the data will only be placed in a low byte buffer and not directly into the low byte timer register. The actual transfer of the data into the low byte timer register is only carried out when a write to its associated high byte timer register, namely TMR1H, is executed. On the other hand, using instructions to preload data into the high byte timer register will result in the data being directly written to the high byte timer register. At the same time the data in the low byte buffer will be transferred into its associated low byte timer register. For this reason, the low byte timer register should be written first when preloading data into the 16-bit timer registers. It must also be noted that to read the contents of the low byte timer register, a read to the high byte timer register must be executed first to latch the contents of the low byte timer register into its associated low byte buffer. After this has been done, the low byte timer register can be read in the normal way. Note that reading the low byte timer register will result in reading the previously latched contents of the low byte buffer and not the actual contents of the low byte timer register.

Timer Control Registers – TMR0C, TMR1C

The flexible features of the Holtek microcontroller Timer/Event Counters enable them to operate in four different modes, the options of which are determined by the contents of their respective control register.

It is the Timer Control Register together with its corresponding timer registers that control the full operation of the Timer/Event Counters. Before the timers can be used, it is essential that the appropriate Timer Control Register is fully programmed with the right data to ensure its correct operation, a process that is normally carried out during program initialisation.

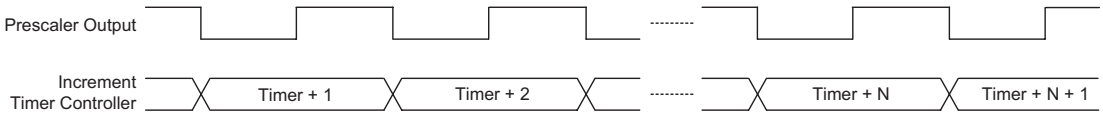
To choose which of the four modes the timer is to operate in, either in the timer mode, the external event counting mode, the internal event counter mode, or the pulse width measurement mode, bits 7 and 6 of the Timer Control Register, which are known as the bit pair T0M1/T0M0 or T1M1/T1M0 respectively, depending upon which timer is used, must be set to the required logic levels. The timer-on bit, which is bit 4 of the Timer Control Register and known as T0ON or T1ON, depending upon which timer is used, provides the basic on/off control of the respective timer. Setting the bit high allows the counter to run, clearing the bit stops the counter. For timers that have prescalers, bits 0~2 of the Timer Control Register determine the division ratio of the input clock prescaler. The prescaler bit settings have no effect if an external clock source is used. If the timer is in the event count or pulse width measurement mode, the active transition edge level type is selected by the logic level of bit 3 of the Timer Control Register which is known as T0E or T1E depending upon which timer is used. An additional T1S bit in the TMR1C register is used to determine the clock source for Timer/Event Counter 1.

Configuring the Timer Mode

In this mode, the Timer/Event Counter can be utilised to measure fixed time intervals, providing an internal interrupt signal each time the Timer/Event Counter overflows. To operate in this mode, the Operating Mode Select bit pair, T0M1/T0M0 or T1M1/T1M0, in the Timer Control Register must be set to the correct value as shown.Control Register Operating Mode Select Bits for the Timer Mode

Bit7	Bit6
1	0

In this mode the internal clock, f_{SYS} , is used as the internal clock for 8-bit Timer/Event Counter 0 and f_{SUB} or $f_{SYS}/4$ is used as the internal clock for 16-bit Timer/Event Counter 1. However, the clock source, f_{SYS}, for 8-bit timer is further divided by a prescaler, the value of which is determined by the Prescaler Rate Select bits T0PSC2~T0PSC0, which are bits 2~0 in the Timer Control Register. After the other bits in the Timer Control Register have been setup, the enable bit T0ON or T1ON, which is bit 4 of the Timer Control Register, can be set high to enable the Timer/Event Counter to run. Each time an internal clock cycle occurs, the Timer/Event Counter increments by one. When it is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bit in the Interrupt Control Register, INTC0, is reset to zero.



Timer Mode Timing Chart

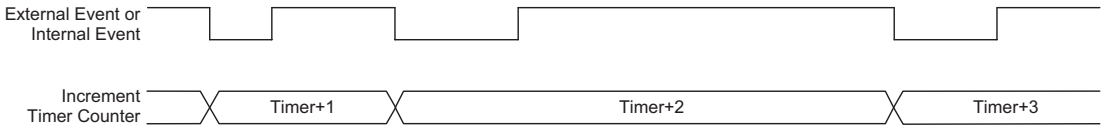
Configuring the Event Counter Mode

In this mode, a number of internally changing logic events, occurring on the internal comparators output, can be recorded by the Timer/Event Counter. To operate in this mode, the Operating Mode Select bit pair, T0M1/T0M0 or T1M1/T1M0, in the Timer Control Register must be set to the correct value as shown. Control Register Operating Mode Select Bits for the Event Counter Mode

Bit7	Bit6
0	0/1

In this mode, the comparator output, CMP1X or CMP2X, is used as the Timer/Event Counter clock source, however it is not divided by the internal prescaler. After the other bits in the Timer Control Register have been setup, the enable bit T0ON or T1ON, which is bit 4 of the Timer Control Register, can be set high to enable the Timer/Event Counter to run. If the Active Edge Select bit T0E or T1E, which is bit 3 of the Timer Control Register, is low, the Timer/Event Counter will increment each time the external timer pin receives a low to high transition. If the Active Edge Select bit is high, the counter will increment each time the external timer pin receives a high to low transition. When it is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bit in the Interrupt Control Register, INTC0, is reset to zero.

It should be noted that in the internal event counting mode, even if the microcontroller is in the Power Down Mode, the Timer/Event Counter will continue to record externally changing logic events on the timer input pin. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



Event Counter Mode Timing Chart (TnE=1)

Configuring the Pulse Width Measurement Mode

In this mode, the Timer/Event Counter can be utilised to measure the width of external pulses applied to the external timer pin. To operate in this mode, the Operating Mode Select bit pair, T0M1/T0M0 or T1M1/T1M0, in the Timer Control Register must be set to the correct value as shown. Control Register Operating Mode Select Bits for the Pulse Width Measurement Mode

Bit7	Bit6
1	1

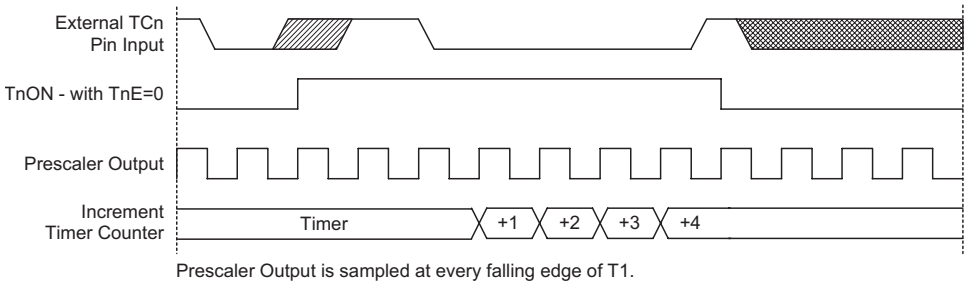
In this mode the internal clock, f_{SYS} , is used as the internal clock for 8-bit Timer/Event Counter 0 and f_{SUB} or $f_{SYS}/4$ is used as the internal clock for 16-bit Timer/Event Counter 1. However, the clock source, f_{SYS}, for 8-bit timer is further divided by a prescaler, the value of which is determined by the Prescaler Rate Select bits T0PSC2~T0PSC0, which are bits 2~0 in the Timer Control Register. After the other bits in the Timer Control Register have been setup, the enable bit T0ON or T1ON, which is bit 4 of the Timer Control Register, can be set high to enable the Timer/Event Counter, however it will not actually start counting until an active edge is received on the external timer pin.

If the Active Edge Select bit T0E or T1E, which is bit 3 of the Timer Control Register, is low, once a high to low transition has been received on the external timer pin, TMR0 or TMR1, the Timer/Event Counter will start counting until the external timer pin returns to its original high level. At this point the enable bit will be automatically reset to zero and the Timer/Event Counter will stop counting. If the Active Edge Select bit is high, the Timer/Event Counter will begin counting once a low to high transition has been received on the external timer pin and stop counting when the external timer pin returns to its original low level. As before, the enable bit will be automatically reset to zero and the Timer/Event Counter will stop counting. It is important to note that in the Pulse Width Measurement Mode, the enable bit is automatically reset to zero when the external control signal on the external timer pin returns to its original level, whereas in the other two modes the enable bit can only be reset to zero under program control.

The residual value in the Timer/Event Counter, which can now be read by the program, therefore represents the length of the pulse received on the external timer pin. As the enable bit has now been reset, any further transitions on the external timer pin will be ignored. Not until the enable bit is again set high by the program can the timer begin further pulse width measurements. In this way, single shot pulse measurements can be easily Made.

It should be noted that in this mode the Timer/Event Counter is controlled by logical transitions on the external timer pin and not by the logic level. When the Timer/Event Counter is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bit in the Interrupt Control Register, INTC0, is reset to zero.

As the external timer pin is shared with an I/O pin, to ensure that the pin is configured to operate as a pulse width measurement pin, two things have to happen. The first is to ensure that the Operating Mode Select bits in the Timer Control Register place the Timer/Event Counter in the Pulse Width Measurement Mode, the second is to ensure that the port control register configures the pin as an input.



Pulse Width Capture Mode Timing Chart (TnE=0)

Programmable Frequency Divider PFD

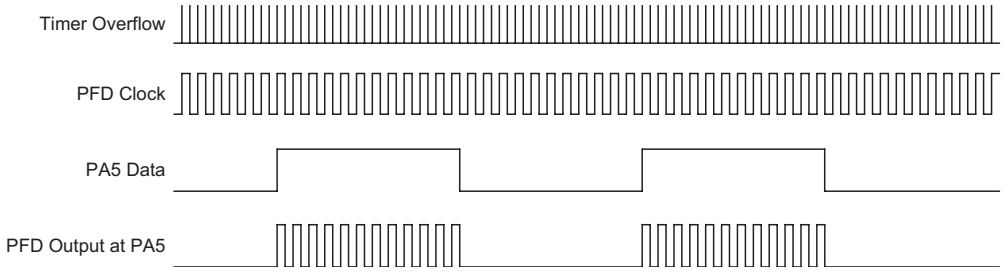
The Programmable Frequency Divider provides a means of producing a variable frequency output suitable for applications requiring a precise frequency generator.

The PFD output is pin-shared with the I/O pin PA5. The PFD function is enabled via PFDEN bit in MISC register, however, if not enabled, the pin can operate as a normal I/O pin.

The clock source for the PFD circuit can originate from either the timer 0 or timer 1 overflow signal selected via PFDSELbit in MISC register. The output frequency is controlled by loading the required values into the timer registers and prescaler registers to give the required division ratio. The timer will begin to count-up from this preload register value until full, at which point an overflow signal is generated, causing the PFD output to change state. The timer will then be automatically reloaded with the preload register value and continue counting-up.

For the PFD output to function, it is essential that the corresponding bit of the Port A control register PAC bit 5 is setup as an output. If setup as an input the PFD output will not function, however, the pin can still be used as a normal input pin. The PFD output will only be activated if bit PA5 is set to "1". This output data bit is used as the on/off control bit for the PFD output. Note that the PFD output will be low if the PA5 output data bit is cleared to "0".

Using this method of frequency generation, and if a crystal oscillator is used for the system clock, very precise values of frequency can be generated.



PFD Output Control

Prescaler

Bits T0PSC0~T0PSC2 of the TMR0C register can be used to define the pre-scaling stages of the internal clock sources of the Timer/Event Counter 0. The Timer/Event Counter overflow signal can be used to generate signals for the PFD and Timer Interrupt.

TMR0C Register

Bit	7	6	5	4	3	2	1	0
Name	T0M1	T0M0	—	T0ON	T0E	T0PSC2	T0PSC1	T0PSC0
R/W	R/W	R/W	—	R/W	R/W	R/W	R/W	R/W
POR	0	0	—	0	1	0	0	0

Bit 7~6 **T0M1, T0M0**: Timer 0 operation mode selection
00: event counter mode, the input signal is from Comparator 1 output
01: event counter mode, the input signal is from TC0 pin
10: timer mode
11: pulse width capture mode

Bit 5 unimplemented, read as "0"

Bit 4 **T0ON**: Timer/Event Counter counting enable
0: disable
1: enable

Bit 3 **T0E**:
Event counter active edge selection
0: count on raising edge
1: count on falling edge
Pulse Width Capture active edge selection
0: start counting on falling edge, stop on rasing edge
1: start counting on raising edge, stop on falling edge

Bit 2~0 **T0PSC2, T0PSC1, T0PSC0**: Timer prescaler rate selection
Timer internal clock=
000: f_{SYS}
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: $f_{SYS}/128$

TMR1C Register

Bit	7	6	5	4	3	2	1	0
Name	T1M1	T1M0	T1S	T1ON	T1E	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	1	—	—	—

Bit 7~6 **T1M1, T1M0**: Timer1 operation mode selection
00: event counter mode, the input signal is from Comparator 2 output
01: event counter mode, the input signal is from TC1 pin
10: timer mode
11: pulse width capture mode

Bit 5 **T1S**: timer clock source
0: $f_{SYS}/4$
1: f_{SUB}, LXT or LIRC

Bit 4 **T1ON**: Timer/event counter counting enable
0: disable
1: enable

Bit 3 **T1E**:
Event counter active edge selection
0: count on raising edge
1: count on falling edge
Pulse width capture active edge selection
0: start counting on falling edge, stop on rasing edge
1: start counting on raising edge, stop on falling edge

Bit 2~0 unimplemented, read as "0"

MISC Register

Bit	7	6	5	4	3	2	1	0
Name	ODE3	ODE2	ODE1	ODE0	—	—	PFDSEL	PFDEN
R/W	R/W	R/W	R/W	R/W	—	—	R/W	R/W
POR	0	0	0	0	—	—	0	0

Bit 7 **ODE3**: PB3 Open Drain Control
0: disable
1: enable

Bit 6 **ODE2**: PB2 Open Drain Control
0: disable
1: enable

Bit 5 **ODE1**: PB1 Open Drain Control
0: disable
1: enable

Bit 4 **ODE0**: PB0 Open Drain Control
0: disable
1: enable

Bit 3~2 Unimplemented, read as "0"

Bit 1 **PFDSEL**: PFD clock selection
0: Timer 0 output
1: Timer 1 output

Bit 0 **PFDEN**: PFD function control
0: PFD disable
1: PFD enable

I/O Interfacing

The Timer/Event Counter, when configured to run in the event counter or pulse width measurement mode, require the use of the external pin for correct operation. As this pin is a shared pin it must be configured correctly to ensure it is setup for use as a Timer/Event Counter input and not as a normal I/O pin. This is implemented by ensuring that the mode select bits in the Timer/Event Counter control register, select either the event counter or pulse width measurement mode. Additionally the Port Control Register must be set high to ensure that the pin is setup as an input. Any pull-high resistor on this pin will remain valid even if the pin is used as a Timer/Event Counter input.

Timer/Event Counter Pins Internal Filter

The external Timer/Event Counter pins are connected to an internal filter to reduce the possibility of unwanted event counting events or inaccurate pulse width measurements due to adverse noise or spikes on the external Timer/Event Counter input signal. As this internal filter circuit will consume a limited amount of power, a configuration option is provided to switch off the filter function, an option which may be beneficial in power sensitive applications, but in which the integrity of the input signal is high. Care must be taken when using the filter on/off configuration option as it will be applied not only to both external Timer/Event Counter pins but also to the external interrupt input pins. Individual Timer/Event Counter or external interrupt pins cannot be selected to have a filter on/off function.

Programming Considerations

When configured to run in the timer mode, the internal system clock is used as the timer clock source and is therefore synchronised with the overall operation of the microcontroller. In this mode when the appropriate timer register is full, the microcontroller will generate an internal interrupt signal directing the program flow to the respective internal interrupt vector. For the pulse width measurement mode, the internal system clock is also used as the timer clock source but the timer will only run when the correct logic condition appears on the external timer input pin. As this is an external event and not synchronized with the internal timer clock, the microcontroller will only see this external event when the next timer clock pulse arrives. As a result, there may be small differences in measured values requiring programmers to take this into account during programming. The same applies if the timer is configured to be in the event counting mode, which again is an external event and not synchronised with the internal system or timer clock.

When the Timer/Event Counter is read, or if data is written to the preload register, the clock is inhibited to avoid errors, however as this may result in a counting error, this should be taken into account by the programmer. Care must be taken to ensure that the timers are properly initialised before using them for the first time. The associated timer enable bits in the interrupt control register must be properly set otherwise the internal interrupt associated with the timer will remain inactive. The edge select, timer mode and clock source control bits in timer control register must also be correctly set to ensure the timer is properly configured for the required application. It is also important to ensure that an initial value is first loaded into the timer registers before the timer is switched on; this is because after power-on the initial values of the timer registers are unknown. After the timer has been initialised the timer can be turned on and off by controlling the enable bit in the timer control register. Note that setting the timer enable bit high to turn the timer on, should only be executed after the timer mode bits have been properly setup. Setting the timer enable bit high together with a mode bit modification, may lead to improper timer operation if executed as a single timer control register byte write instruction.

When the Timer/Event counter overflows, its corresponding interrupt request flag in the interrupt control register will be set. If the timer interrupt is enabled this will in turn generate an interrupt signal. However irrespective of whether the interrupts are enabled or not, a Timer/Event counter overflow will also generate a wake-up signal if the device is in a Power-down condition. This situation may occur if the Timer/Event Counter is in the Event Counting Mode and if the external signal continues to change state. In such a case, the Timer/Event Counter will continue to count these external events and if an overflow occurs the device will be woken up from its Power-down condition. To prevent such a wake-up from occurring, the timer interrupt request flag should first be set high before issuing the HALT instruction to enter the Power Down Mode.

Timer Program Example

This program example shows how the Timer/Event Counter registers are setup, along with how the interrupts are enabled and managed. Note how the Timer/Event Counter is turned on, by setting bit 4 of the Timer Control Register. The Timer/Event Counter can be turned off in a similar way by clearing the same bit. This example program sets the Timer/Event Counter to be in the timer mode, which uses the internal system clock as the clock source.

```
org 04h             ; external interrupt vector

org 0ch             ; Timer/Event Counter 0 interrupt vector
jmp tmrint          ; jump here when the Timer/Event Counter 0 overflows
:
org 20h             ; main program
                    ; internal Timer/Event Counter 0 interrupt routine
:
tmrint:
                    ; Timer/Event Counter 0 main program placed here
:

reti:
:

begin:
                    ; setup Timer 0 registers
mov a,09bh          ; setup Timer 0 preload value
mov tmr0,a;
mov a,081h          ; setup Timer 0 control register
mov tmr0c,a         ; timer mode and prescaler set to /2
                    ; setup interrupt register
mov a,009h          ; enable master interrupt and timer interrupt
mov intc0,a
set tmr0c.4         ; start Timer/Event Counter 0 - note mode bits must be previously
                    ; setup
```

Pulse Width Modulator

The HT45F23A contains two channels of 8-bit PWM function. Useful for such applications such as motor speed control, the PWM function provides outputs with a fixed frequency but with a duty cycle that can be varied by setting particular values into the corresponding PWM register.

PWM Operation

A single register, known as PWMn and located in the Data Memory is assigned to each Pulse Width Modulator channel. It is here that the 8-bit value, which represents the overall duty cycle of one modulation cycle of the output waveform, should be placed. To increase the PWM modulation frequency, each modulation cycle is subdivided into two or four individual modulation subsections, known as the 7+1 mode or 6+2 mode respectively. The required mode and the on/off control for each PWM channel is selected using the BPCTL register. Note that when using the PWM, it is only necessary to write the required value into the PWMn register and select the required mode setup and on/off control using the BPCTL registers, the subdivision of the waveform into its sub-modulation cycles is implemented automatically within the microcontroller hardware. The PWM clock source is the system clock f_{SYS}. This method of dividing the original modulation cycle into a further 2 or 4 sub-cycles enable the generation of higher PWM frequencies which allow a wider range of applications to be served. The difference between what is known as the PWM cycle frequency and the PWM modulation frequency should be understood. As the PWM clock is the system clock, f_{SYS}, and as the PWM value is 8-bits wide, the overall PWM cycle frequency is $f_{SYS}/256$. However, when in the 7+1 mode of operation the PWM modulation frequency will be $f_{SYS}/128$, while the PWM modulation frequency for the 6+2 mode of operation will be $f_{SYS}/64$.

PWM Modulation	PWM Cycle Frequency	PWM Cycle Duty
$f_{SYS}/64$ for (6+2) bits mode $f_{SYS}/128$for (7+1) bits mode	$f_{SYS}/256$	[PWM]/256

BPCTL Register

Bit	7	6	5	4	3	2	1	0
Name	PMODE	PWM1EN	PWM0EN	BC1	BC0	BZ2	BZ1	BZ0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **PMODE**: PWM type selection
0: 7+1
1: 6+2

Bit 6 **PWM1EN**: PWM1 or the other pin-shared functions
0: the other pin-shared functions
1: PWM1

Bit 5 **PWM0EN**: PWM0 or the other pin-shared functions
0: the other pin-shared functions
1: PWM0

Bit 4~0 Buzzer output and I/O configuration selection, described elsewhere

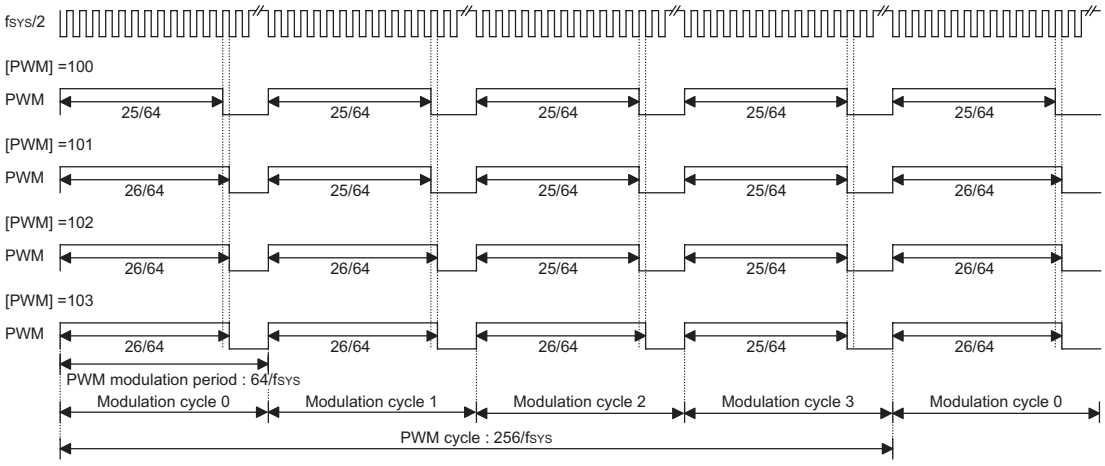
6+2 PWM Mode

Each full PWM cycle, as it is controlled by an 8-bit PWM register, has 256 clock periods. However, in the 6+2 PWM mode, each PWM cycle is subdivided into four individual sub-cycles known as modulation cycle 0 ~ modulation cycle 3, denoted as i in the table. Each one of these four sub-cycles contains 64 clock cycles. In this mode, a modulation frequency increase of four is achieved. The 8-bit PWM register value, which represents the overall duty cycle of the PWM waveform, is divided into two groups. The first group which consists of bit2~bit7 is denoted here as the DC value. The second group which consists of bit0~bit1 is known as the AC value. In the 6+2 PWM mode, the duty cycle value of each of the four modulation sub-cycles is shown in the following table.

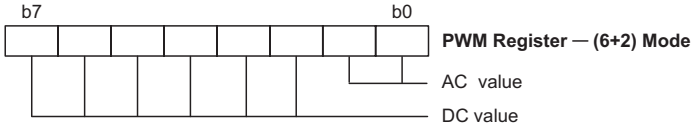
Parameter	AC (0~3)	DC (Duty Cycle)
Modulation cycle i (i=0~3)	i<AC	$\frac{DC+1}{64}$
	i≥AC	$\frac{DC}{64}$

6+2 Mode Modulation Cycle Values

The following diagram illustrates the waveforms associated with the 6+2 mode of PWM operation. It is important to note how the single PWM cycle is subdivided into 4 individual modulation cycles, numbered from 0~3 and how the AC value is related to the PWM value.



6+2 PWM Mode



PWM Register for 6+2 Mode

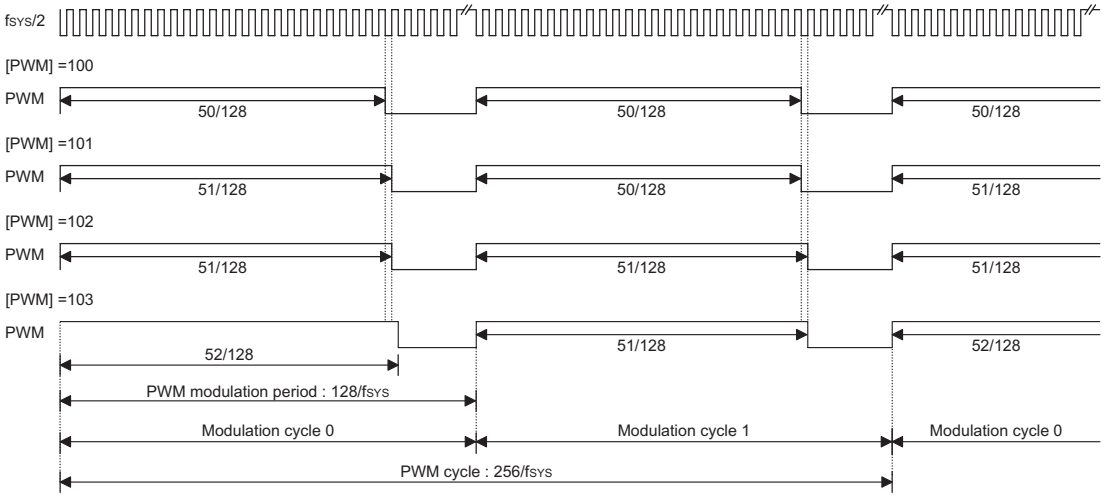
7+1 PWM Mode

Each full PWM cycle, as it is controlled by an 8-bit PWM register, has 256 clock periods. However, in the 7+1 PWM mode, each PWM cycle is subdivided into two individual sub-cycles known as modulation cycle 0 ~ modulation cycle 1, denoted as i in the table. Each one of these two sub-cycles contains 128 clock cycles. In this mode, a modulation frequency increase of two is achieved. The 8-bit PWM register value, which represents the overall duty cycle of the PWM waveform, is divided into two groups. The first group which consists of bit1~bit7 is denoted here as the DC value. The second group which consists of bit0 is known as the AC value. In the 7+1 PWM mode, the duty cycle value of each of the two modulation sub-cycles is shown in the following table.

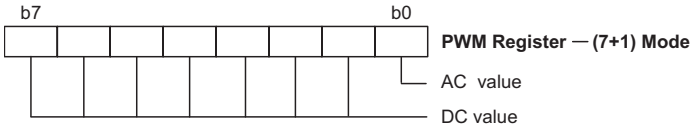
Parameter	AC (0~1)	DC (Duty Cycle)
Modulation cycle i (i=0~1)	i<AC	$\frac{DC+1}{128}$
	i≥AC	$\frac{DC}{128}$

7+1 Mode Modulation Cycle Values

The following diagram illustrates the waveforms associated with the 7+1 mode PWM operation. It is important to note how the single PWM cycle is subdivided into 2 individual modulation cycles, numbered 0 and 1 and how the AC value is related to the PWM value.



7+1 PWM Mode



PWM Register for 7+1 Mode

PWM Output Control

The PWM outputs are pin-shared with the I/O pins PC5 and PC6. To operate as a PWM output and not as an I/O pin, the correct bits must be set in the BPCTL register. A high value must be written to the PWM0EN or PWM1EN to select the corresponding PWM. A zero value must also be written to the corresponding bit in the I/O port control register PCC.5 and PCC.6 to ensure that the corresponding PWM output pin is setup as an output. After these two initial steps have been carried out, and of course after the required PWM value has been written into the PWMn register, writing a high value to the corresponding bit in the output data register PC.5 and PC.6 will enable the PWM data to appear on the pin. Writing a zero value will disable the PWM output function and force the output low. In this way, the Port data output registers can be used as an on/off control for the PWM function. Note that if the BPCTL register has selected the PWM function, but a high value has been written to its corresponding bit in the PCC control register to configure the pin as an input, then the pin can still function as a normal input line, with pull-high resistor options.

PWM Programming Example

The following sample program shows how the PWM0 output is setup and controlled.

```
mov a,64h          ;setup PWM value of decimal 100
mov pwm0,a
clr bpctl.7        ;select the 7+1 PWM mode
set bpctl.5        ;select PWM0
clr pcc.5          ;setup pin PC5
set pc.5           ;enable the PWM output
: :
clr pc.5           ; disable the PWM output pin,PC5 forced low
```

Analog to Digital Converter

The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the microcontroller, the need for external components is reduced significantly with the corresponding follow-on benefits of lower costs and reduced component space requirements.

A/D Overview

The HT45F23A contains a multi-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into either a 12-bit digital value.

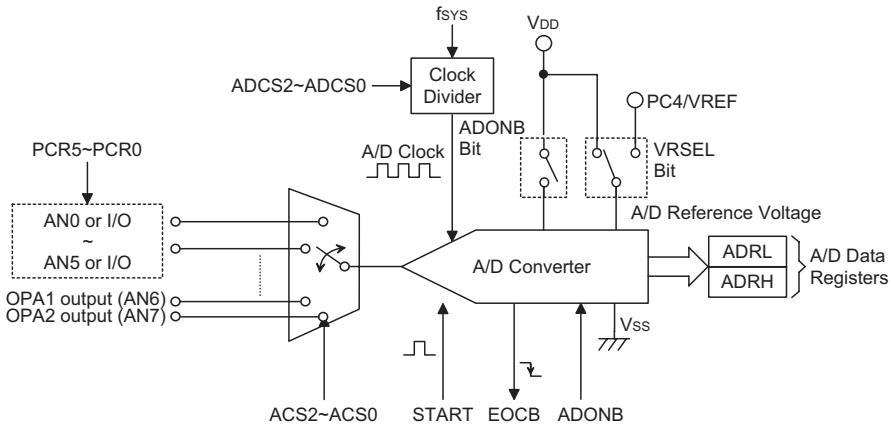
The accompanying block diagram shows the overall internal structure of the A/D converter, together with its associated registers.

A/D Converter Register Description

Overall operation of the A/D converter is controlled using five registers. A read only register pair exists to store the ADC data 12-bit value. The remaining three registers are control registers which setup the operating and control function of the A/D converter.

Register Name	Bit							
	7	6	5	4	3	2	1	0
ADCR	START	EOCB	—	—	—	ACS2	ACS1	ACS0
ACSR	—	ADONB	VRSEL	—	—	ADCS2	ADCS1	ADCS0
ADPCR	—	—	PCR5	PCR4	PCR3	PCR2	PCR1	PCR0

A/D Converter Register List



A/D Converter Structure

A/D Converter Data Registers – ADRL, ADRH

As the device contain an internal 12-bit A/D converter, they require two data registers to store the converted value. These are a high byte register, known as ADRH, and a low byte register, known as ADRL. After the conversion process takes place, these registers can be directly read by the microcontroller to obtain the digitised conversion value. D0~D11 are the A/D conversion result data bits. Any unused bits will be read as zero.

ADRH								ADRL							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0

A/D Data Registers

A/D Converter Control Registers – ADCR, ACSR, ADPCR

To control the function and operation of the A/D converter, three control registers known as ADCR, ACSR and ADPCR are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter end of conversion status. The ACS2~ACS0 bits in the ADCR register define the ADC input channel number. As the device contains only one actual analog to digital converter hardware circuit, each of the individual 8 analog inputs, which include 6 external A/D channels and 2 internal OPA outputs, must be routed to the converter. It is the function of the ACS2~ACS0 bits to determine which analog channel input pin is actually connected to the internal A/D converter.

The ADPCR control register contains the PCR5~PCR0 bits which determine which pins on Port B and Port C are used as analog inputs for the A/D converter input and which pins are not to be used as the A/D converter input. Setting the corresponding bit high will select the A/D input function, clearing the bit to zero will select either the I/O or other pin-shared function. When the pin is selected to be an A/D input, its original function whether it is an I/O or other pin-shared function will be removed. In addition, any internal pull-high resistors connected to these pins will be automatically removed if the pin is selected to be an A/D input.

ADCR Register

Bit	7	6	5	4	3	2	1	0
Name	START	EOCB	—	—	—	ACS2	ACS1	ACS0
R/W	R/W	R	—	—	—	R/W	R/W	R/W
POR	0	1	—	—	—	0	0	0

Bit 7 **START**: Start the A/D conversion
0→1→0 : start
0→1 : reset the A/D converter and set EOCB to "1"
This bit is used to initiate an A/D conversion process. The bit is normally low but if set high and then cleared low again, the A/D converter will initiate a conversion process. When the bit is set high the A/D converter will be reset.

Bit 6 **EOCB**: End of A/D conversion flag
0: A/D conversion ended
1: A/D conversion in progress
This read only flag is used to indicate when an A/D conversion process has completed. When the conversion process is running the bit will be high.

Bit 5~3 unimplemented, read as "0"

Bit 2~0 **ACS2, ACS1, ACS0**: Select A/D channel
000: AN0
001: AN1
010: AN2
011: AN3
100: AN4
101: AN5
110: AN6, connect Op Amp 1 output (A1E)
111: AN7, connect Op Amp 2 output (A2E)
These are the A/D channel select control bits. As there is only one internal hardware A/D converter each of the eight A/D inputs must be routed to the internal converter using these bits.

ACSR Register

Bit	7	6	5	4	3	2	1	0
Name	—	ADONB	VRSEL	—	—	ADCS2	ADCS1	ADCS0
R/W	—	R/W	R/W	—	—	R/W	R/W	R/W
POR	1	0	0	—	—	0	0	0

Bit 7 unimplemented, read as "1"

Bit 6 **ADONB**: ADC module power on/off control bit
0: ADC module power on
1: ADC module power off
This bit controls the power to the A/D internal function. This bit should be cleared to zero to enable the A/D converter. If the bit is set high then the A/D converter will be switched off reducing the device power consumption. As the A/D converter will consume a limited amount of power,even when not executing a conversion, this may be an important consideration in power sensitive battery powered applications.
Note: 1. it is recommended to set ADONB=1 before entering IDLE/SLEEP Mode for saving power.
2. ADONB=1 will power down the ADC module.

Bit 5 **VRSEL**: Selecte ADC reference voltage
0: Internal ADC power
1: VREF pin or LDO output (2.4V/3.3V)
This bit is used to select the reference voltage for the A/D converter. If the bit is high, then the A/D converter reference voltage is supplied on the external VREF pin or LDO output (2.4V/3.3V).If the pin is low, then the internal reference is used which is taken from the power supply pin VDD.

Bit 4~3 unimplemented, read as "0"

Bit 2~0 **ADCS2, ADCS1, ADCS0**: Select ADC clock source
000: $f_{SYS}/2$
001: $f_{SYS}/8$
010: $f_{SYS}/32$
011: Undefined
100: f_{SYS}
101: $f_{SYS}/4$
110: $f_{SYS}/16$
111: Undefined
These three bits are used to select the clock source for the A/D converter.

ADPCR Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	PCR5	PCR4	PCR3	PCR2	PCR1	PCR0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 unimplemented, read as "0"

Bit 5 **PCR5**: Define PC1 is A/D input or not
0: Not A/D input
1: A/D input, AN5

Bit 4 **PCR4**: Define PC0 is A/D input or not
0: Not A/D input
1: A/D input, AN4

Bit 3 **PCR3**: Define PB6 is A/D input or not
0: Not A/D input
1: A/D input, AN3

Bit 2 **PCR2**: Define PB5 is A/D input or not
0: Not A/D input
1: A/D input, AN2

Bit 1 **PCR1**: Define PB4 is A/D input or not
0: Not A/D input
1: A/D input, AN1

Bit 0 **PCR0**: Define PB3 is A/D input or not
0: Not A/D input
1: A/D input, AN0

A/D Operation

The START bit in the ADCR register is used to start and reset the A/D converter. When the microcontroller sets this bit from low to high and then low again, an analog to digital conversion cycle will be initiated. When the START bit is brought from low to high but not low again, the EOCB bit in the ADCR register will be set high and the analog to digital converter will be reset. It is the START bit that is used to control the overall start operation of the internal analog to digital converter.

The EOCB bit in the ADCR register is used to indicate when the analog to digital conversion process is complete. This bit will be automatically set to "0" by the microcontroller after a conversion cycle has ended. In addition, the corresponding A/D interrupt request flag will be set in the interrupt control register, and if the interrupts are enabled, an appropriate internal interrupt signal will be generated. This A/D internal interrupt signal will direct the program flow to the associated A/D internal interrupt address for processing. If the A/D internal interrupt is disabled, the microcontroller can be used to poll the EOCB bit in the ADCR register to check whether it has been cleared as an alternative method of detecting the end of an A/D conversion cycle.

The clock source for the A/D converter, which originates from the system clock f_{SYS}, can be chosen to be either f_{SYS} or a subdivided version of f_{SYS}. The division ratio value is determined by the ADCS2~ADCS0 bits in the ACSR register.

Although the A/D clock source is determined by the system clock f_{SYS}, and by bits ADCS2~ADCS0, there are some limitations on the maximum A/D clock source speed that can be selected. As the minimum value of permissible A/D clock period, t_{ADCK}, is 0.5μs, care must be taken for system clock frequencies equal to or greater than 4MHz. For example, if the system clock operates at a frequency of 4MHz, the ADCS2~ADCS0 bits should not be set to "000". Doing so will give A/D clock periods that are less than the minimum A/D clock period which may result in inaccurate A/D conversion values. Refer to the following table for examples, where values marked with an asterisk * show where, depending upon the device, special care must be taken, as the values may be less than the specified minimum A/D Clock Period.

f_{SYS}	A/D Clock Period (t_{ADCK})						
	ADCS2, ADCS1, ADCS0 = 100 (f_{SYS})	ADCS2, ADCS1, ADCS0 = 000 (f_{SYS}/2)	ADCS2, ADCS1, ADCS0 = 101 (f_{SYS}/4)	ADCS2, ADCS1, ADCS0 = 001 (f_{SYS}/8)	ADCS2, ADCS1, ADCS0 = 110 (f_{SYS}/16)	ADCS2, ADCS1, ADCS0 = 010 (f_{SYS}/32)	ADCS2, ADCS1, ADCS0 = 111 = 011
1MHz	1μs	2μs	4μs	8μs	16μs	32μs	Undefined
2MHz	500ns	1μs	2μs	4μs	8μs	16μs	Undefined
4MHz	250ns*	500ns	1μs	2μs	4μs	8μs	Undefined
8MHz	125ns*	250ns*	500ns	1μs	2μs	4μs	Undefined
12MHz	83ns*	167ns*	333ns*	667ns	1.33μs	2.67μs	Undefined

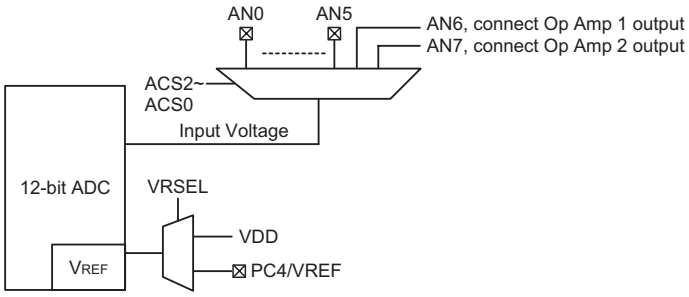
A/D Clock Period Examples

Controlling the power on/off function of the A/D converter circuitry is implemented using the ADONB bit in the ACSR register. This bit must be zero to power on the A/D converter. When the ADONB bit is cleared to zero to power on the A/D converter internal circuitry a certain delay, as indicated in the timing diagram, must be allowed before an A/D conversion is initiated. Even if no pins are selected for use as A/D inputs by clearing the PCR5~PCR0 bits in the ADPCR register, if the ADONB bit is zero then some power will still be consumed. In power conscious applications it is therefore recommended that the ADONB is set high to reduce power consumption when the A/D converter function is not being used.

The reference voltage supply to the A/D Converter can be supplied from either the positive power supply pin, VDD, or from an external reference sources supplied on pin VREF. The desired selection is made using the VRSEL bit. As the VREF pin is pin-shared with other functions, when the VRSEL bit is set high, the VREF pin function will be selected and the other pin functions will be disabled automatically.

A/D Input Pins

The A/D analog input pins are pin-shared with the I/O pins on Port B and Port C as well as other functions. The PCR5~ PCR0 bits in the ADPCR register, determine whether the input pins are setup as A/D converter analog inputs or whether they have other functions. If the PCR5~ PCR0 bits for its corresponding pin is set high then the pin will be setup to be an A/D converter input and the original pin functions disabled. In this way, pins can be changed under program control to change their function between A/D inputs and other functions. All pull-high resistors, which are setup through register programming, will be automatically disconnected if the pins are setup as A/D inputs. Note that it is not necessary to first setup the A/D pin as an input in the PBC or PCC port control register to enable the A/D input as when the PCR5~ PCR0 bits enable an A/D input, the status of the port control register will be overridden.



A/D Input Structure

The A/D converter has its own reference voltage pin, VREF, however the reference voltage can also be supplied from the power supply pin, a choice which is made through the VRSEL bit in the ACSR register. The analog input values must not be allowed to exceed the value of VREF.

Summary of A/D Conversion Steps

The following summarises the individual steps that should be executed in order to implement an A/D conversion process.

- Step 1
  Select the required A/D conversion clock by correctly programming bits ADCS2~ADCS0 in the ACSR register.
- Step 2
  Enable the A/D by clearing the ADONB bit in the ACSR register to zero.
- Step 3
  Select which channel is to be connected to the internal A/D converter by correctly programming the ACS2~ACS0 bits which are also contained in the ADCR register.
- Step 4
  Select which pins are to be used as A/D inputs and configure them by correctly programming the PCR5~PCR0 bits in the ADPCR register.

- Step 5
  If the interrupts are to be used, the interrupt control registers must be correctly configured to ensure the A/D converter interrupt function is active. The master interrupt control bit, EMI, Mulit-function interrupt bit, and the A/D converter interrupt bit, EADI, must all be set high to do this.
- Step 6
  The analog to digital conversion process can now be initialised by setting the START bit in the ADCR register from low to high and then low again. Note that this bit should have been originally cleared to zero.
- Step 7
  To check when the analog to digital conversion process is complete, the EOCB bit in the ADCR register can be polled. The conversion process is complete when this bit goes low. When this occurs the A/D data registers ADRL and ADRH can be read to obtain the conversion value. As an alternative method, if the interrupts are enabled and the stack is not full, the program can wait for an A/D interrupt to occur.

Note: When checking for the end of the conversion process, if the method of polling the EOCB bit in the ADCR register is used, the interrupt enable step above can be omitted.

The accompanying diagram shows graphically the various stages involved in an analog to digital conversion process and its associated timing. After an A/D conversion process has been initiated by the application program, the microcontroller internal hardware will begin to carry out the conversion, during which time the program can continue with other functions. The time taken for the A/D conversion is $16t_{ADCK}$ where t_{ADCK} is equal to the A/D clock period.

Programming Considerations

During microcontroller operations where the A/D converter is not being used, the A/D internal circuitry can be switched off to reduce power consumption, by setting bit ADONB high in the ACSR register. When this happens, the internal A/D converter circuits will not consume power irrespective of what analog voltage is applied to their input lines. If the A/D converter input lines are used as normal I/Os, then care must be taken as if the input voltage is not at a valid logic level, then this may lead to some increase in power consumption.

A/D Transfer Function

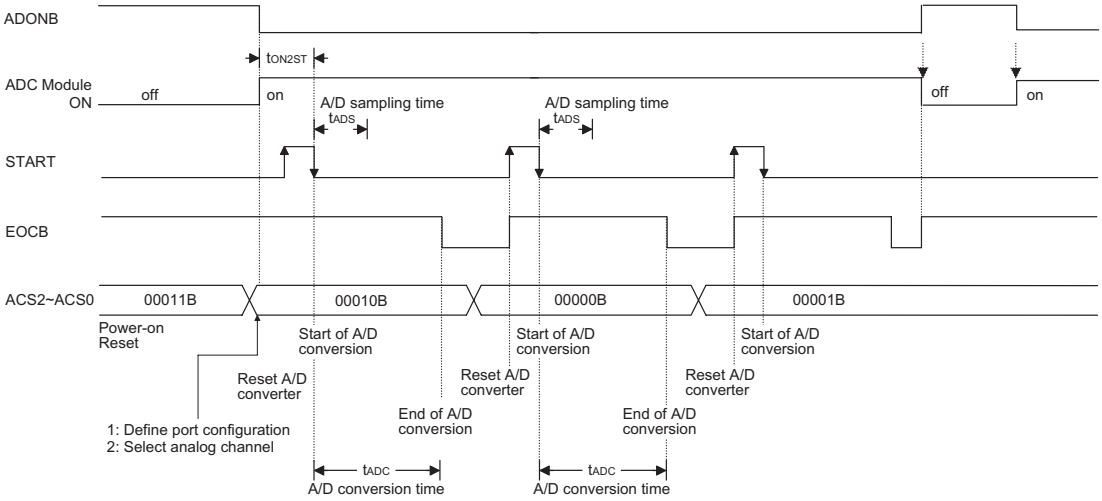
As the device contain a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to the V_{DD} or V_{REF} voltage, this gives a single bit analog input value of V_{DD} or V_{REF} divided by 4096.

$$1\ LSB = (V_{DD}\ or\ V_{REF}) \div 4096$$

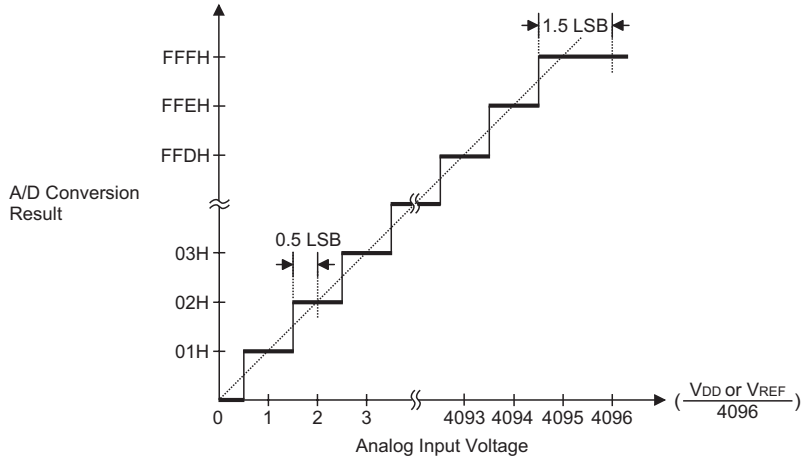
The A/D Converter input voltage value can be calculated using the following equation:

$$A/D\ input\ voltage = A/D\ output\ digital\ value \times (V_{DD}\ or\ V_{REF}) \div 4096$$

The diagram shows the ideal transfer function between the analog input value and the digitised output value for the A/D converter. Except for the digitised zero value, the subsequent digitised values will change at a point 0.5 LSB below where they would change without the offset, and the last full scale digitised value will change at a point 1.5 LSB below the V_{DD} or V_{REF} level.



A/D Conversion Timing



Ideal A/D Transfer Function

A/D Programming Example

The following two programming examples illustrate how to setup and implement an A/D conversion. In the first example, the method of polling the EOCB bit in the ADCR register is used to detect when the conversion cycle is complete, whereas in the second example, the A/D interrupt is used to determine when the conversion is complete.

- Example: using an EOCB polling method to detect the end of conversion

```
    clr EADI                ; disable ADC interrupt
    mov a,01H
    mov ACSR,a              ; select fSYS/8 as A/D clock
                            ; Select VDD as ADC reference voltage and turn on ADONB bit
    mov a,0Fh               ; setup ADPCR to configure pins AN0~AN3
    mov ADPCR,a
    mov a,00h
    mov ADCR,a              ; enable and connect AN0 channel to A/D converter
    :
start_conversion:
    clr START               ; high pulse on start bit to initiate conversion
    set START               ; reset A/D
    clr START               ; start A/D
polling_EOC:
    sz EOCB                 ; poll the ADCR0 register EOCB bit to detect end of A/D conversion
    jmp polling_EOC         ; continue polling
    mov a,ADRL              ; read low byte conversion result value
    mov ADRL_buffer,a       ; save result to user defined register
    mov a,ADRH              ; read high byte conversion result value
    mov ADRH_buffer,a       ; save result to user defined register
    :
    :
    jmp start_conversion    ; start next a/d conversion
```

- Example: using the interrupt method to detect the end of conversion

```
    clr  EADI            ; disable ADC interrupt
    mov a,01H
    mov AcsR,a           ; select fSYS/8 as A/D clock
                         ; select VDD as ADC reference voltage and turn on
                         ; ADONB bit
    mov a,0Fh            ; setup ADPCR to configure pins AN0~AN3
    mov ADPCR,a
    mov a,00h
    mov ADCR,a           ; enable and connect AN0 channel to A/D converter
Start_conversion:
    clr START            ; high pulse on START bit to initiate conversion
    set START            ; reset A/D
    clr START            ; start A/D
    clr ADF              ; clear ADC interrupt request flag
    set EADI             ; enable ADC interrupt
    set EMFI             ; enable Multi-function interrupt
    set EMI              ; enable global interrupt
    :
    :
                         ; ADC interrupt service routine
ADC_:
    mov acc_stack,a      ; save ACC to user defined memory
    mov a,STATUS
    mov status_stack,a   ; save STATUS to user defined memory
    :
    :
    mov a,ADRL           ; read low byte conversion result value
    mov adrl_buffer,a    ; save result to user defined register
    mov a,ADRH           ; read high byte conversion result value
    mov adrh_buffer,a    ; save result to user defined register
    :
    :
EXIT_INT_ISR:
    mov a,status_stack
    mov STATUS,a         ; restore STATUS from user defined memory
    mov a,acc_stack      ; restore ACC from user defined memory
    Clr ADF              ; clear ADC interrupt request flag
    reti
```

Serial Interface Module – SIM

The HT45F23A contains a Serial Interface Module, which includes both the four line SPI interface or the two line I²C interface types, to allow an easy method of communication with external peripheral hardware. Having relatively simple communication protocols, these serial interface types allow the microcontroller to interface to external SPI or I²C based hardware such as sensors, Flash or EEPROM memory, etc. The SIM interface pins are pin-shared with PB0~PB3 pins therefore the SIM interface function must first be selected using a configuration option. As both interface types share the same pins and registers, the choice of whether the SPI or I²C type is used is made using the SIM operating mode control bits, named SIM2~SIM0, in the SIMC0 register. These pull-high resistors of the SIM pin-shared I/O are selected using pull-high control registers, and also if the SIM function is enabled.

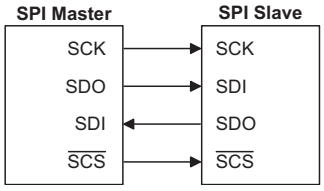
SPI Interface

The SPI interface is often used to communicate with external peripheral device such as sensors, Flash or EEPROM memory device etc. Originally developed by Motorola, the four line SPI interface is a synchronous serial data interface that has a relatively simple communication protocol simplifying the programming requirements when communicating with external hardware device.

The communication is full duplex and operates as a slave/master type, where the device can be either master or slave. Although the SPI interface specification can control multiple slave device from a single master, but this device provided only one $\overline{SCS}$ pin. If the master needs to control multiple slave devices from a single master, the master can use I/O pin to select the slave devices.

SPI Interface Operation

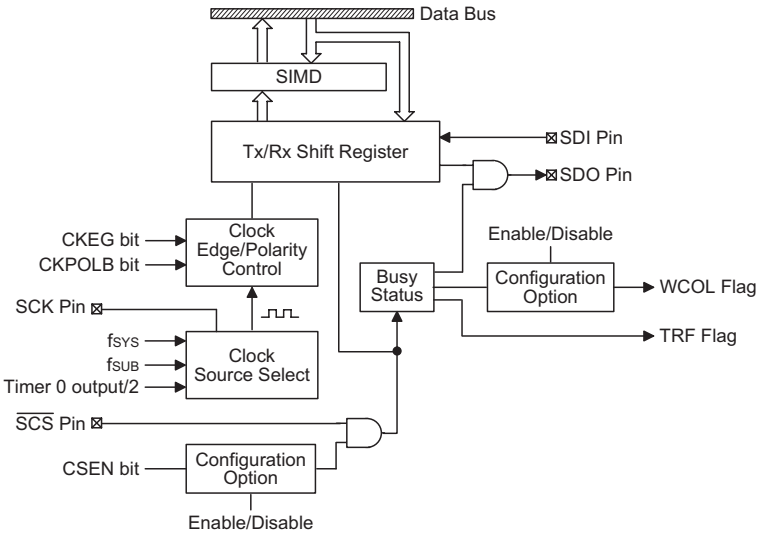
The SPI interface is a full duplex synchronous serial data link. It is a four line interface with pin names SDI, SDO, SCK and $\overline{SCS}$. Pins SDI and SDO are the Serial Data Input and Serial Data Output lines, SCK is the Serial Clock line and $\overline{SCS}$ is the Slave Select line. As the SPI interface pins are pin-shared with PB0~PB3 pins and with the I²C function pins, the SPI interface must first be enabled by selecting the SIM enable configuration option and setting the correct bits in the SIMC0 and SIMC2 registers. After the SPI configuration option has been configured it can also be additionally disabled or enabled using the SIMEN bit in the SIMC0 register. Communication between devices connected to the SPI interface is carried out in a slave/master mode with all data transfer initiations being implemented by the master. The Master also controls the clock signal. As the device only contains a single $\overline{SCS}$ pin only one slave device can be utilized. The $\overline{SCS}$ pin is controlled by software, set CSEN bit to "1" to enable $\overline{SCS}$ pin function, set CSEN bit to "0" the $\overline{SCS}$ pin will be floating state.



SPI Master/Slave Connection

The SPI function in this device offers the following features:

- Full duplex synchronous data transfer
- Both Master and Slave modes
- LSB first or MSB first data transmission modes
- Transmission complete flag
- Rising or falling active clock edge
- WCOL and CSEN bit enabled or disable select



The status of the SPI interface pins is determined by a number of factors such as whether the device is in the master or slave mode and upon the condition of certain control bits such as CSEN and SIMEN.

There are several configuration options associated with the SPI interface. One of these is to enable the SIM function which selects the SIM pins rather than normal I/O pins. Note that if the configuration option does not select the SIM function then the SIMEN bit in the SIMC0 register will have no effect. Another two SPI configuration options determine if the CSEN and WCOL bits are to be used.

SPI Registers

There are three internal registers which control the overall operation of the SPI interface. These are the SIMD data register and two registers SIMC0 and SIMC2. Note that the SIMC1 register is only used by the I²C interface.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	PCKEN	PCKP1	PCKP0	SIMEN	—
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMC2	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF

SIM Registers List

The SIMD register is used to store the data being transmitted and received. The same register is used by both the SPI and I²C functions. Before the device writes data to the SPI bus, the actual data to be transmitted must be placed in the SIMD register. After the data is received from the SPI bus, the device can read it from the SIMD register. Any transmission or reception of data from the SPI bus must be made via the SIMD register.

SIMD Register

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

"x" unknown

There are also two control registers for the SPI interface, SIMC0 and SIMC2. Note that the SIMC2 register also has the name SIMA which is used by the I²C function. The SIMC1 register is not used by the SPI function, only by the I²C function. Register SIMC0 is used to control the enable/disable function and to set the data transmission clock frequency. Although not connected with the SPI function, the SIMC0 register is also used to control the Peripheral Clock Prescaler. Register SIMC2 is used for other control functions such as LSB/MSB selection, write collision flag etc.

SIMC0 Register

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	PCKEN	PCKP1	PCKP0	SIMEN	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	1	1	1	0	0	0	0	—

Bit 7~5 **SIM2, SIM1, SIM0**: SIM Operating Mode Control
000: SPI master mode; SPI clock is $f_{SYS}/4$
001: SPI master mode; SPI clock is $f_{SYS}/16$
010: SPI master mode; SPI clock is $f_{SYS}/64$
011: SPI master mode; SPI clock is f_{SUB}
100: SPI master mode; SPI clock is Timer 0 output/2 (PFD0)
101: SPI slave mode
110: I²C slave mode
111: Unused mode
These bits setup the overall operating mode of the SIM function. As well as selecting if the I²C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from the Timer 0. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 **PCKEN**: PCK Output Pin Control
0: Disable
1: Enable

Bit 3~2 **PCKP1, PCKP0**: Select PCK output pin frequency
00: f_{SYS}
01: $f_{SYS}/4$
10: $f_{SYS}/8$
11: Timer 0 output/2 (PFD0)

Bit 1 **SIMEN**: SIM Control
0: Disable
1: Enable
The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to zero to disable the SIM interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will be in a floating condition and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. The SIM configuration option must have first enabled the SIM interface for this bit to be effective. If the SIM is configured to operate as an SPI interface via the SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM is configured to operate as an I²C interface via the SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I²C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I²C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 unimplemented, read as "0"

SIMC2 Register

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 Undefined bit
This bit can be read or written by user software program.

Bit 5 **CKPOLB**: Determines the base condition of the clock line
0: the SCK line will be high when the clock is inactive
1: the SCK line will be low when the clock is inactive
The CKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCK line will be low when the clock is inactive. When the CKPOLB bit is low, then the SCK line will be high when the clock is inactive.

Bit 4 **CKEG**: Determines SPI SCK active clock edge type
CKPOLB=0
0: SCK is high base level and data capture at SCK rising edge
1: SCK is high base level and data capture at SCK falling edge
CKPOLB=1
0: SCK is low base level and data capture at SCK falling edge
1: SCK is low base level and data capture at SCK rising edge
The CKEG and CKPOLB bits are used to setup the way that the clock signal outputs and inputs data on the SPI bus. These two bits must be configured before data transfer is executed otherwise an erroneous clock edge may be generated. The CKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCK line will be low when the clock is inactive. When the CKPOLB bit is low, then the SCK line will be high when the clock is inactive. The CKEG bit determines active clock edge type which depends upon the condition of CKPOLB bit.

Bit 3 **MLS**: SPI Data shift order
0: LSB
1: MSB
This is the data shift select bit and is used to select how the data is transferred, either MSB or LSB first. Setting the bit high will select MSB first and low for LSB first.

Bit 2 **CSEN**: SPI $\overline{SCS}$ pin Control
0: Disable
1: Enable
The CSEN bit is used as an enable/disable for the $\overline{SCS}$ pin. If this bit is low, then the $\overline{SCS}$ pin will be disabled and placed into a floating condition. If the bit is high the $\overline{SCS}$ pin will be enabled and used as a select pin. Note that using the CSEN bit can be disabled or enabled via configuration option.

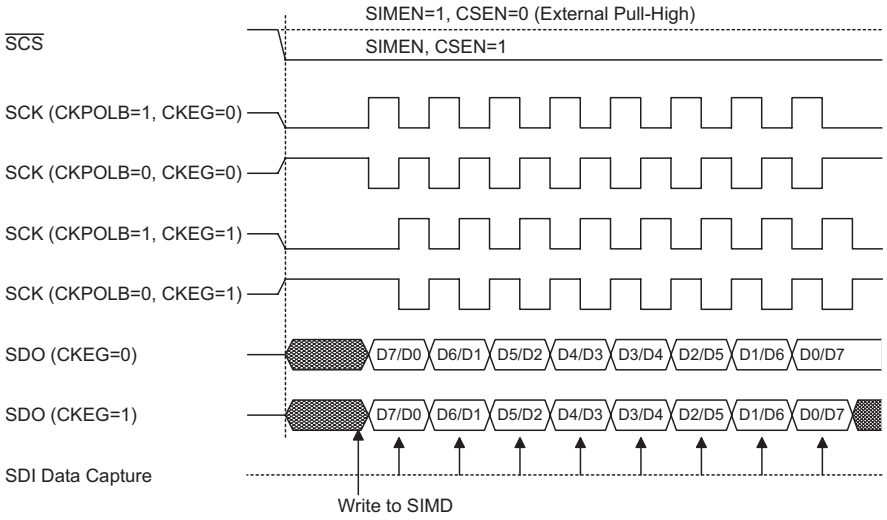
Bit 1 **WCOL**: SPI Write Collision flag
0: No collision
1: Collision
The WCOL flag is used to detect if a data collision has occurred. If this bit is high it means that data has been attempted to be written to the SIMD register during a data transfer operation. This writing operation will be ignored if data is being transferred. The bit can be cleared by the application program. Note that using the WCOL bit can be disabled or enabled via configuration option.

Bit 0 **TRF**: SPI Transmit/Receive Complete flag
0: Data is being transferred
1: SPI data transmission is completed
The TRF bit is the Transmit/Receive Complete flag and is set "1" automatically when an SPI data transmission is completed, but must set to "0" by the application program. It can be used to generate an interrupt.

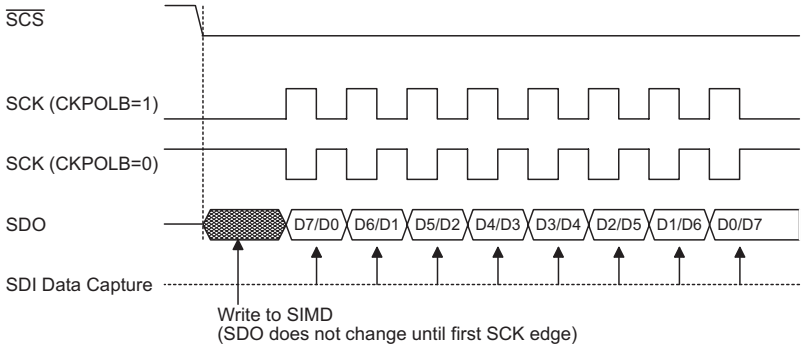
SPI Communication

After the SPI interface is enabled by setting the SIMEN bit high, then in the Master Mode, when data is written to the SIMD register, transmission/reception will begin simultaneously. When the data transfer is complete, the TRF flag will be set automatically, but must be cleared using the application program. In the Slave Mode, when the clock signal from the master has been received, any data in the SIMD register will be transmitted and any data on the SDI pin will be shifted into the SIMD register. The master should output an $\overline{SCS}$ signal to enable the slave device before a clock signal is provided. The slave data to be transferred should be well prepared at the appropriate moment relative to the $\overline{SCS}$ signal depending upon the configurations of the CKPOLB bit and CKEG bit. The accompanying timing diagram shows the relationship between the slave data and $\overline{SCS}$ signal for various configurations of the CKPOLB and CKEG bits.

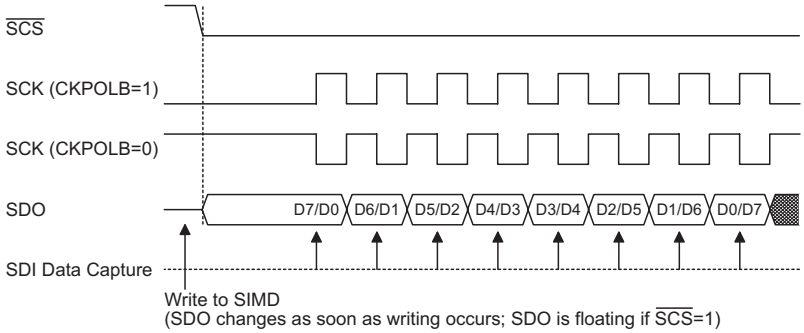
The SPI will continue to function even in the IDLE Mode.



SPI Master Mode Timing

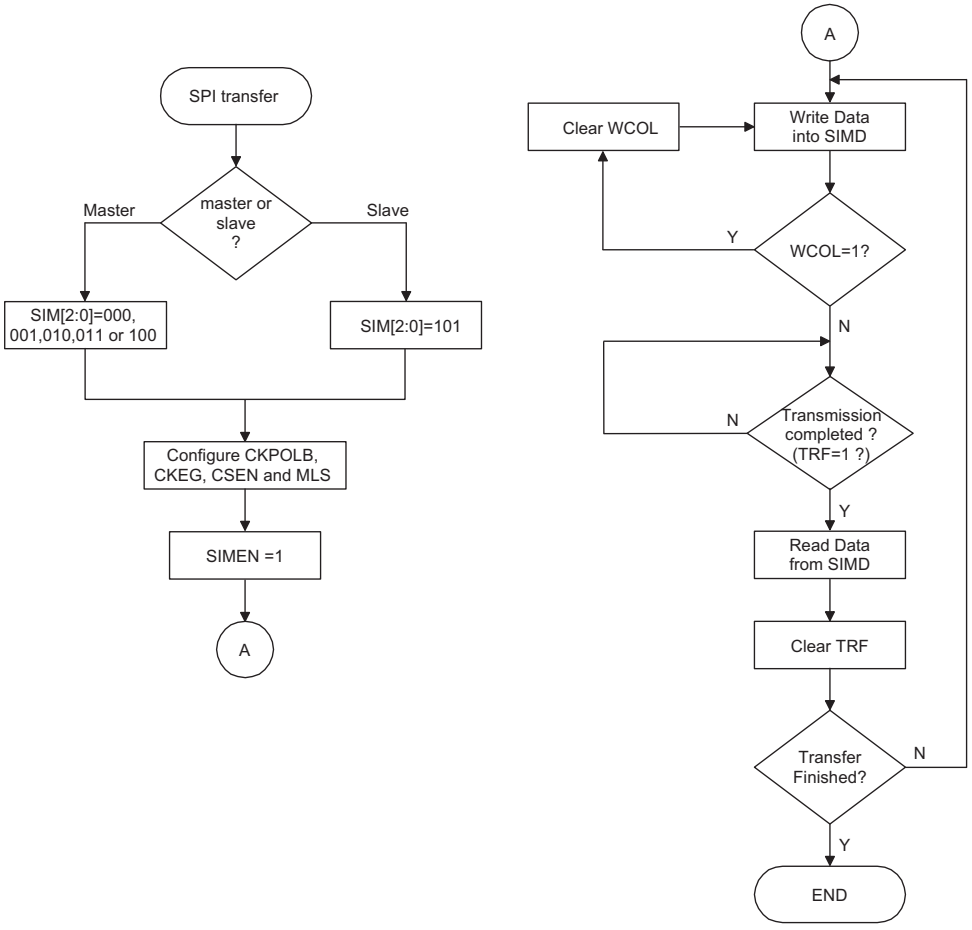


SPI Slave Mode Timing-CKEG=0



Note: For SPI slave mode, if SIMEN=1 and CSEN=0, SPI is always enabled and ignores the $\overline{SCS}$ level.

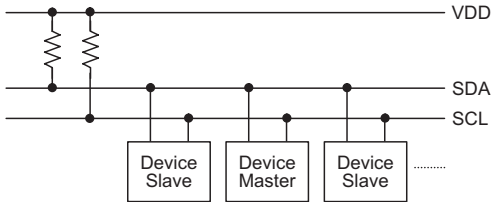
SPI Slave Mode Timing-CKEG=1



SPI Transfer Control Flowchart

I²C Interface

The I²C interface is used to communicate with external peripheral device such as sensors, EEPROM memory etc. Originally developed by Philips, it is a two line low speed serial interface for synchronous serial data transfer. The advantage of only two lines for communication, relatively simple communication protocol and the ability to accommodate multiple devices on the same bus has made it an extremely popular interface type for many applications.



I²C Master Slave Bus Connection

I²C Interface Operation

The I²C serial interface is a two line interface, a serial data line, SDA, and serial clock line, SCL. As many devices may be connected together on the same bus, their outputs are both open drain types. For this reason it is necessary that external pull-high resistors are connected to these outputs. Note that no chip select line exists, as each device on the I²C bus is identified by a unique address which will be transmitted and received on the I²C bus.

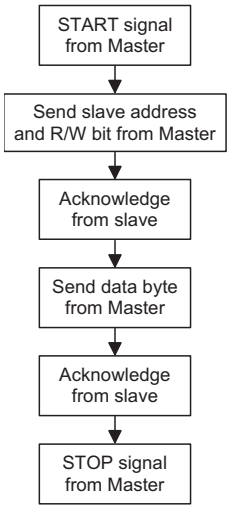
When two devices communicate with each other on the bidirectional I²C bus, one is known as the master device and one as the slave device. Both master and slave can transmit and receive data, however, it is the master device that has overall control of the bus. For these devices, which only operates in slave mode, there are two methods of transferring data on the I²C bus, the slave transmit mode and the slave receive mode.

There are several configuration options associated with the I²C interface. One of these is to enable the function which selects the SIM pins rather than normal I/O pins. Note that if the configuration option does not select the SIM function then the SIMEN bit in the SIMC0 register will have no effect. A configuration option exists to allow a clock other than the system clock to drive the I²C interface. Another configuration option determines the debounce time of the I²C interface. This uses the internal clock to in effect add a debounce time to the external clock to reduce the possibility of glitches on the clock line causing erroneous operation. The debounce time, if selected, can be chosen to be either 1 or 2 system clocks.

I²C Registers

There are three control registers associated with the I²C bus, SIMC0, SIMC1 and SIMA and one data register, SIMD. The SIMD register, which is shown in the above SPI section, is used to store the data being transmitted and received on the I²C bus. Before the microcontroller writes data to the I²C bus, the actual data to be transmitted must be placed in the SIMD register. After the data is received from the I²C bus, the microcontroller can read it from the SIMD register. Any transmission or reception of data from the I²C bus must be made via the SIMD register.

Note that the SIMA register also has the name SIMC2 which is used by the SPI function. Bit SIMEN and bits SIM2~SIM0 in register SIMC0 are used by the I²C interface.



Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	PCKEN	PCKP1	PCKP0	SIMEN	—
SIMC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMA	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	D0

I²C Registers List

SIMC0 Register

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	PCKEN	PCKP1	PCKP0	SIMEN	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	1	1	1	0	0	0	0	—

Bit 7~5 **SIM2, SIM1, SIM0**: Operating Mode Control
000: SPI master mode; SPI clock is $f_{SYS}/4$
001: SPI master mode; SPI clock is $f_{SYS}/16$
010: SPI master mode; SPI clock is $f_{SYS}/64$
011: SPI master mode; SPI clock is f_{SUB}
100: SPI master mode; SPI clock is Timer 0 output/2 (PFD0)
101: SPI slave mode
110: I²C slave mode
111: Unused mode

These bits setup the overall operating mode of the SIM function. As well as selecting if the I²C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from the Timer 0. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 **PCKEN**: PCK Output Pin Control
0: Disable
1: Enable

Bit 3~2 **PCKP1, PCKP0**: Select PCK output pin frequency
00: f_{SYS}
01: $f_{SYS}/4$
10: $f_{SYS}/8$
11: Timer 0 output/2 (PFD0)

Bit 1 **SIMEN**: SIM Control
0: Disable
1: Enable
The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to zero to disable the SIM interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will be in a floating condition and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. The SIM configuration option must have first enabled the SIM interface for this bit to be effective. If the SIM is configured to operate as an SPI interface via SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM is configured to operate as an I^2C interface via the SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I^2C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I^2C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 unimplemented, read as "0"

SIMC1 Register

Bit	7	6	5	4	3	2	1	0
Name	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
R/W	R	R	R	R/W	R/W	R	R/W	R
POR	1	0	0	0	0	0	0	1

Bit 7 **HCF**: I^2C Bus data transfer completion flag
0: Data is being transferred
1: Completion of an 8-bit data transfer
The HCF flag is the data transfer flag. This flag will be zero when data is being transferred. Upon completion of an 8-bit data transfer the flag will go high and an interrupt will be generated.

Bit 6 **HAAS**: I^2C Bus address match flag
0: Not address match
1: Address match
The HASS flag is the address match flag. This flag is used to determine if the slave device address is the same as the master transmit address. If the addresses match then this bit will be high, if there is no match then the flag will be low.

Bit 5 **HBB**: I^2C Bus busy flag
0: I^2C Bus is not busy
1: I^2C Bus is busy
The HBB flag is the I^2C busy flag. This flag will be "1" when the I^2C bus is busy which will occur when a START signal is detected. The flag will be set to "0" when the bus is free which will occur when a STOP signal is detected.

Bit 4 **HTX**: Select I^2C slave device is transmitter or receiver
0: Slave device is the receiver
1: Slave device is the transmitter

Bit 3 **TXAK**: I^2C Bus transmit acknowledge flag
0: Slave send acknowledge flag
1: Slave do not send acknowledge flag
The TXAK bit is the transmit acknowledge flag. After the slave device receipt of 8-bits of data, this bit will be transmitted to the bus on the 9th clock from the slave device. The slave device must always set TXAK bit to "0" before further data is received.

Bit 2 **SRW**: I^2C Slave Read/Write flag
0: Slave device should be in receive mode
1: Slave device should be in transmit mode

The SRW flag is the I²C Slave Read/Write flag. This flag determines whether the master device wishes to transmit or receive data from the I²C bus. When the transmitted address and slave address is match, that is when the HAAS flag is set high, the slave device will check the SRW flag to determine whether it should be in transmit mode or receive mode. If the SRW flag is high, the master is requesting to read data from the bus, so the slave device should be in transmit mode. When the SRW flag is zero, the master will write data to the bus, therefore the slave device should be in receive mode to read this data.

Bit 1 **IAMWU**: I²C Address Match Wake-up Control
0: Disable
1: Enable
This bit should be set to "1" to enable I²C address match wake up from SLEEP or IDLE Mode.

Bit 0 **RXAK**: I²C Bus Receive acknowledge flag
0: Slave receive acknowledge flag
1: Slave do not receive acknowledge flag
The RXAK flag is the receiver acknowledge flag. When the RXAK flag is "0", it means that a acknowledge signal has been received at the 9th clock, after 8 bits of data have been transmitted. When the slave device in the transmit mode, the slave device checks the RXAK flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the RXAK flag is "1". When this occurs, the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I²C Bus.

The SIMD register is used to store the data being transmitted and received. The same register is used by both the SPI and I²C functions. Before the device writes data to the SPI bus, the actual data to be transmitted must be placed in the SIMD register. After the data is received from the SPI bus, the device can read it from the SIMD register. Any transmission or reception of data from the SPI bus must be made via the SIMD register.

SIMD Register

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

"x" unknown

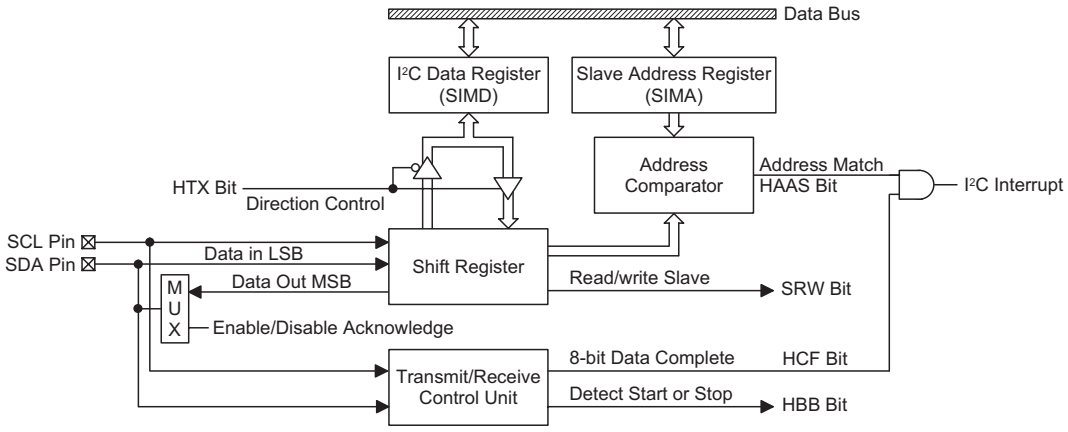
SIMA Register

Bit	7	6	5	4	3	2	1	0
Name	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	x	x	x	x	x	x	x	—

"x" unknown

Bit 7~1 IICA6~ IICA0: I²C slave address
IICA6~ IICA0 is the I²C slave address bit 6~ bit 0.
The SIMA register is also used by the SPI interface but has the name SIMC2. The SIMA register is the location where the 7-bit slave address of the slave device is stored. Bits 7~ 1 of the SIMA register define the device slave address. Bit 0 is not defined.
When a master device, which is connected to the I²C bus, sends out an address, which matches the slave address in the SIMA register, the slave device will be selected. Note that the SIMA register is the same register address as SIMC2 which is used by the SPI interface.

Bit 0 Undefined bit
This bit can be read or written by user software program.

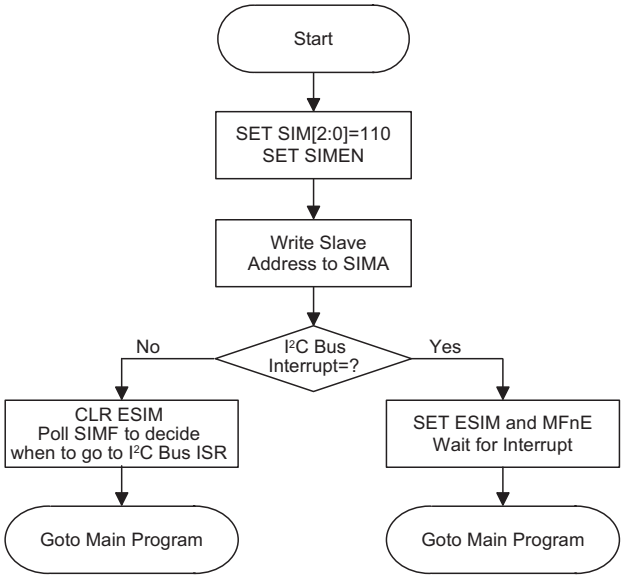


I²C Block Diagram

I²C Bus Communication

Communication on the I²C bus requires four separate steps, a START signal, a slave device address transmission, a data transmission and finally a STOP signal. When a START signal is placed on the I²C bus, all devices on the bus will receive this signal and be notified of the imminent arrival of data on the bus. The first seven bits of the data will be the slave address with the first bit being the MSB. If the address of the slave device matches that of the transmitted address, the HAAS bit in the SIMC1 register will be set and an I²C interrupt will be generated. After entering the interrupt service routine, the slave device must first check the condition of the HAAS bit to determine whether the interrupt source originates from an address match or from the completion of an 8-bit data transfer. During a data transfer, note that after the 7-bit slave address has been transmitted, the following bit, which is the 8th bit, is the read/write bit whose value will be placed in the SRW bit. This bit will be checked by the slave device to determine whether to go into transmit or receive mode. Before any transfer of data to or from the I²C bus, the microcontroller must initialise the bus, the following are steps to achieve this:

- Step 1
  Set the SIM2~SIM0 and SIMEN bits in the SIMC0 register to "1" to enable the I²C bus.
- Step 2
  Write the slave address of the device to the I²C bus address register SIMA.
- Step 3
  Set the ESIM and SIM Muti-Function interrupt enable bit of the interrupt control register to enable the SIM interrupt and Multi-function interrupt.



I²C Bus Initialisation Flow Chart

I²C Bus Start Signal

The START signal can only be generated by the master device connected to the I²C bus and not by the slave device. This START signal will be detected by all devices connected to the I²C bus. When detected, this indicates that the I²C bus is busy and therefore the HBB bit will be set. A START condition occurs when a high to low transition on the SDA line takes place when the SCL line remains high.

Slave Address

The transmission of a START signal by the master will be detected by all devices on the I²C bus. To determine which slave device the master wishes to communicate with, the address of the slave device will be sent out immediately following the START signal. All slave devices, after receiving this 7-bit address data, will compare it with their own 7-bit slave address. If the address sent out by the master matches the internal address of the microcontroller slave device, then an internal I²C bus interrupt signal will be generated. The next bit following the address, which is the 8th bit, defines the read/write status and will be saved to the SRW bit of the SIMC1 register. The slave device will then transmit an acknowledge bit, which is a low level, as the 9th bit. The slave device will also set the status flag HAAS when the addresses match.

As an I²C bus interrupt can come from two sources, when the program enters the interrupt subroutine, the HAAS bit should be examined to see whether the interrupt source has come from a matching slave address or from the completion of a data byte transfer. When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.

I²C Bus Read/Write Signal

The SRW bit in the SIMC1 register defines whether the slave device wishes to read data from the I²C bus or write data to the I²C bus. The slave device should examine this bit to determine if it is to be a transmitter or a receiver. If the SRW flag is "1" then this indicates that the master device wishes to read data from the I²C bus, therefore the slave device must be setup to send data to the I²C bus as a transmitter. If the SRW flag is "0" then this indicates that the master wishes to send data to the I²C bus, therefore the slave device must be setup to read data from the I²C bus as a receiver.

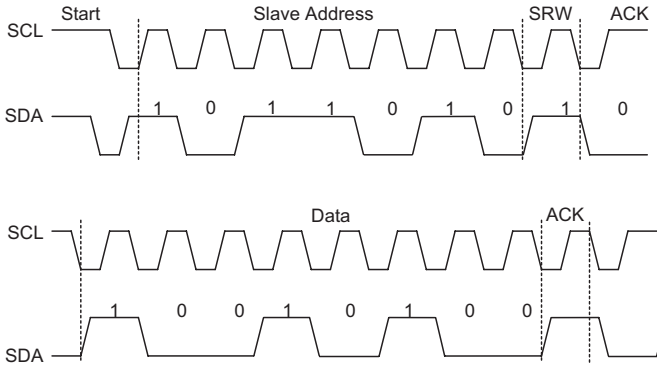
I²C Bus Slave Address Acknowledge Signal

After the master has transmitted a calling address, any slave device on the I²C bus, whose own internal address matches the calling address, must generate an acknowledge signal. The acknowledge signal will inform the master that a slave device has accepted its calling address. If no acknowledge signal is received by the master then a STOP signal must be transmitted by the master to end the communication. When the HAAS flag is high, the addresses have matched and the slave device must check the SRW flag to determine if it is to be a transmitter or a receiver. If the SRW flag is high, the slave device should be setup to be a transmitter so the HTX bit in the SIMC1 register should be set to "1". If the SRW flag is low, then the microcontroller slave device should be setup as a receiver and the HTX bit in the SIMC1 register should be set to "0".

I²C Bus Data and Acknowledge Signal

The transmitted data is 8-bits wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial bit transmission is the MSB first and the LSB last. After receipt of 8-bits of data, the receiver must transmit an acknowledge signal, level "0", before it can receive the next data byte. If the slave transmitter does not receive an acknowledge bit signal from the master receiver, then the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I²C Bus. The corresponding data will be stored in the SIMD register. If setup as a transmitter, the slave device must first write the data to be transmitted into the SIMD register. If setup as a receiver, the slave device must read the transmitted data from the SIMD register.

When the slave receiver receives the data byte, it must generate an acknowledge bit, known as TXAK, on the 9th clock. The slave device, which is setup as a transmitter will check the RXAK bit in the SIMC1 register to determine if it is to send another data byte, if not then it will release the SDA line and await the receipt of a STOP signal from the master.

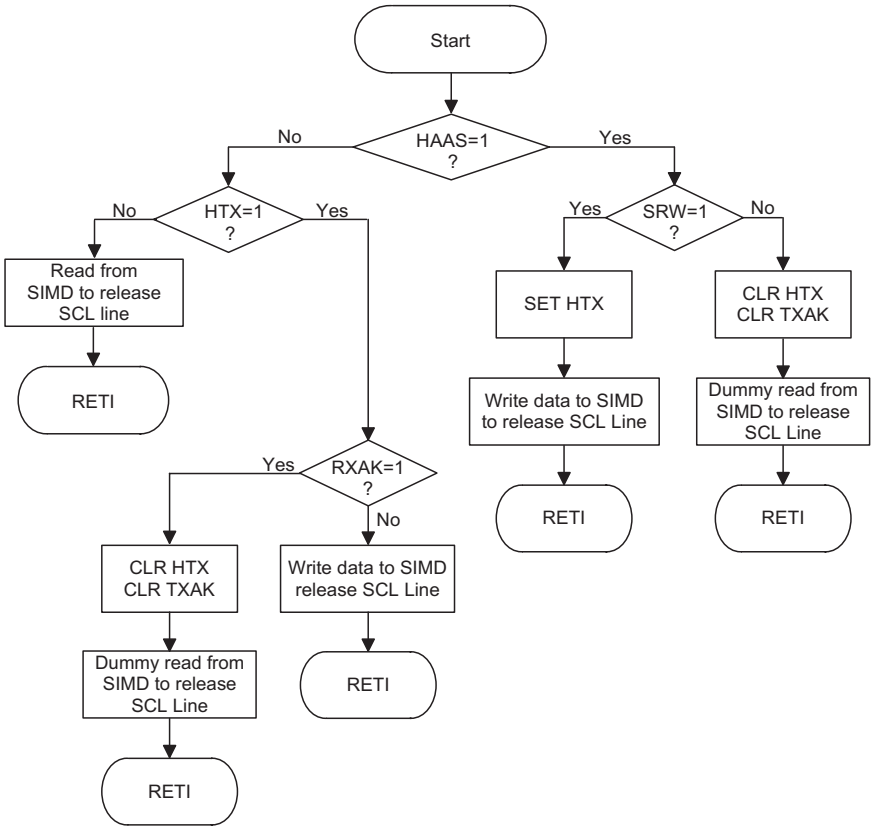


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver 1 bit)
P=Stop (1 bit)

S	SA	SR	M	D	A	D	A	--------	S	SA	SR	M	D	A	D	A	--------	P

Note: * When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.

I²C Communication Timing Diagram



I²C Bus ISR Flow Chart

Peripheral Clock Output

The Peripheral Clock Output allows the device to supply external hardware with a clock signal synchronised to the microcontroller clock.

Peripheral Clock Operation

As the peripheral clock output pin, PCK, is shared with PB4, the required pin function is chosen via PCKEN in the SIMC0 register. The Peripheral Clock function is controlled using the SIMC0 register. The clock source for the Peripheral Clock Output can originate from either the Timer 0 output/2 or a divided ratio of the internal f_{SYS} clock. The PCKEN bit in the SIMC0 register is the overall on/off control, setting PCKEN bit to "1" enables the Peripheral Clock, setting PCKEN bit to "0" disables it. The required division ratio of the system clock is selected using the PCKP1 and PCKP0 bits in the same register. If the device enters the SLEEP Mode this will disable the Peripheral Clock output.

SIMC0 Register

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	PCKEN	PCKP1	PCKP0	SIMEN	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	1	1	1	0	0	0	0	—

Bit 7~5 **SIM2, SIM1, SIM0**: SIM operating mode control
000: SPI master mode; SPI clock is $f_{SYS}/4$
001: SPI master mode; SPI clock is $f_{SYS}/16$
010: SPI master mode; SPI clock is $f_{SYS}/64$
011: SPI master mode; SPI clock is f_{SUB}
100: SPI master mode; SPI clock is Timer 0 output/2 (PFD0)
101: SPI slave mode
110: I^2C slave mode
111: Unused mode
These bits setup the overall operating mode of the SIM function. As well as selecting if the I^2C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from the Timer 0. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 **PCKEN**: PCK output pin control
0: Disable
1: Enable

Bit 3~2 **PCKP1, PCKP0**: select PCK output pin frequency
00: f_{SYS}
01: $f_{SYS}/4$
10: $f_{SYS}/8$
11: Timer 0 output/2 (PFD0)

Bit 1 **SIMEN**: SIM control
0: Disable
1: Enable
The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to zero to disable the SIM interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will be in a floating condition and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. The SIM configuration option must have first enabled the SIM interface for this bit to be effective. Note that when the SIMEN bit changes from low to high the contents of the SPI control registers will be in an unknown condition and should therefore be first initialised by the application program.

Bit 0 unimplemented, read as "0"

SCOM Function for LCD

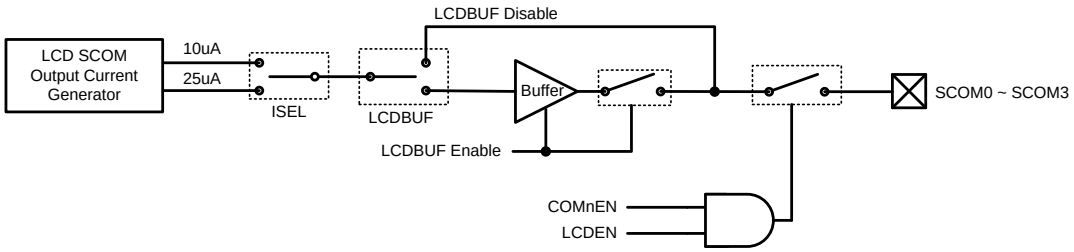
The device have the capability of driving external LCD panels. The common pins for LCD driving, SCOM0~SCOM3, are pin shared with certain pin on the PA0, PC4~PC6 pins. The LCD signals (COM and SEG) are generated using the application program.

LCD Operation

An external LCD panel can be driven using this device by configuring the PA0 and PC4~PC6 pins as common pins and using other output ports lines as segment pins. The LCD driver function is controlled using the LCDC register which in addition to controlling the overall on/off function also controls the bias voltage setup function. This enables the LCD COM driver to generate the necessary VDD/2 voltage levels for LCD 1/2 bias operation.

The LCDEN bit in the LCDC register is the overall master control for the LCD driver, however this bit is used in conjunction with the COMnEN bits to select which I/O pins are used for LCD driving. Note that the Port Control register does not need to first setup the pins as outputs to enable the LCD driver operation.

The following block diagram illustrates the functional structure for LCD COM function.



HT45F23A LCD Circuit

LCDEN	COMnEN	Pin Function	Output Level
0	X	I/O	High or Low
1	0	I/O	High or Low
1	1	SCOMn	VM

Output Control

LCD Bias Control

The LCD COM driver enables two kinds of selection to be provided to suit the requirement of the LCD panel which is being used. The bias resistor choice is implemented using the ISEL bit in the LCDC register.

LCDC Register

Bit	7	6	5	4	3	2	1	0
Name	—	LCDBUF	ISEL	LCDEN	COM3EN	COM2EN	COM1EN	COM0EN
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 unimplemented, read as "0"

Bit 6 **LCDBUF**: LCD buffer control bit
0: disable
1: enable

Bit 5 **ISEL**: SCOM operating current selection (V_{DD}=5V)
0: 10μA
1: 25μA

Bit 4 **LCDEN**: LCD control bit
0: disable
1: enable
The SCOMn can be enable by COMnEN if LCDEN=1.

Bit 3 **COM3EN**: PC6 or SCOM3 selection
0: GPIO
1: SCOM3

Bit 2 **COM2EN**: PC5 or SCOM2 selection
0: GPIO
1: SCOM2

Bit 1 **COM1EN**: PC4 or SCOM1 selection
0: GPIO
1: SCOM1

Bit 0 **COM0EN**: PA0 or SCOM0 selection
0: GPIO
1: SCOM0

Note: This device provides the LCD buffer function, which is controlled by LCDBUF flag, to prevent the interference from LCD panel. With this buffer, that will provide more stable reference voltages, VH0/1, VL0/1, for OPA and Comparator. It should be noted that if the LCD SCOM power supply is selected from VLDO or if the LCD panel has larger size, than the LCD buffer should be turned on to have a higher driver current. However, that will cause more power consumption to turn on this buffer.

LDO Function

The device contain a low power voltage regulator implemented in CMOS technology. Using CMOS technology ensures low voltage drop and low quiescent current. There are two fixed output voltages of 2.4V and 3.3V, which can be controlled by a specific register. The internal LDO output combined with various options by register can provide a fixed voltage for the LCD bias voltage, the OPA reference voltage, the ADC reference voltage and as a fixed power supply for external device.

LDOC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	VLOE	REN1	VRES	VSEL	LDOEN
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

Bit 7~5 unimplemented, read as "0"

Bit 4 **VLOE**: LDO output voltage control bit
0: disable
1: enable
If the VLOE and LDOEN are set to "1", the LDO will output 2.4V or 3.3V to pin and disable I/O function.

Bit 3 **REN1**: Bias voltage divided resistor control bit
0: disable
1: enable
If the REN1is set to "1", that will turn on the resistor DC path, which will generate bias voltage for OPAs or LCD SCOM.

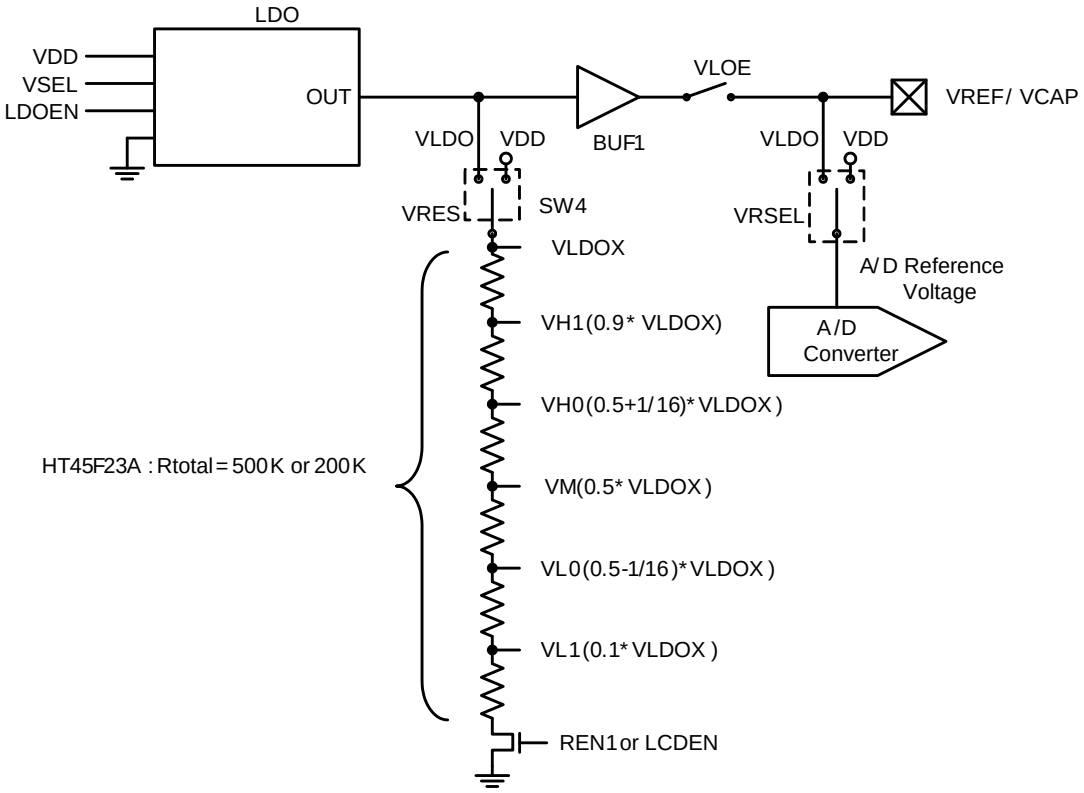
Bit 2 **VRES**: Divided resistor voltage supply selection bit
0: VDD
1: VLDO

Bit 1 **VSEL**: LDO output voltage selection bit
0: 2.4V
1: 3.3V

Bit 0 **LDOEN**: LDO control bit
0: disable
1: enable

Note: 1. The total resistance of the divided resistor, 500K or 200K, can be selected by the ISEL flag in LCDC register.
2. To disable the LDO function will turn off the BUF1 as well, no matter the LDO output voltage control bit, VLOE, is enabled or not.
3. If the LDO output is as the ADC reference voltage, then the VCAP should be connected a 0.1μF capacitor to ground.
4. If the LDO is disabled, LDOEN=0, then the SW4 will be turned to VDD, no matter VRES flag is "1" or not.

The following block diagram illustrates the functional structure for LDO and divided resister.



Operational Amplifiers

There are two fully integrated Operational Amplifiers in the device, OPA1 and OPA2. These OPAs can be used for signal amplification according to specific user requirements. The OPAs can be disabled or enabled entirely under software control using internal registers. With specific control registers, some OPA related applications can be more flexible and easier to be implemented, such as Unit Gain Buffer, Non-Inverting Amplifier, Inverting Amplifier and various kinds of filters, etc.

Operational Amplifier Registers

The internal Operational Amplifiers are fully under the control of internal registers, OPA1C0, OPA1C1, OPA2C0, OPA2C1 and OPA2C2. These registers control enable/disable function, input path selection, gain control and polarity.

OPA1C0 Register

Bit	7	6	5	4	3	2	1	0
Name	A1X	—	—	—	—	—	—	—
R/W	R	—	—	—	—	—	—	—
POR	0	—	—	—	—	—	—	—

Bit 7 **A1X**: Operational amplifier output; positive logic. This bit is read only.

Bit 6~0 unimplemented, read as "0"

OPA1C1 Register

Bit	7	6	5	4	3	2	1	0
Name	A1O2CIN	A1O2N	A1PSEL1	A1PSEL0	A1PS	A1NS	A1OEN	A1EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	1	1	0	0

Bit 7 **A1O2CIN**: OPA1 output to comparator input control bit
0: disable
1: enable

Bit 6 **A1O2N**: OPA1 output to OPA1 Inverting input control bit
0: disable
1: enable

Bit 5~4 **A1PSEL1, A1PSEL0**: OPA1 Non-inverting input selection bit
00: no connection
01: from VH1 (0.9×VLDO)
10: from VM (0.5×VDD or 0.5×VLDO)
11: from VL1 (0.1×VDD or 0.1×VLDO)

Bit 3 **A1PS**: A1P pin to OPA1 Non-inverting input control bit
0: no connection
1: from A1P pin

Bit 2 **A1NS**: A1N pin to OPA1 Inverting input control bit
0: no connection
1: from A1N pin

Bit 1 **A1OEN**: OPA1 output enable or disable control bit
0: disable
1: enable
Note: If OPA1 enable and A1OEN set to 1, the MCU will consumption more DC power (100uA ~ 200uA).

Bit 0 **A1EN**: OPA1 enable or disable control bit
0: disable
1: enable

OPA2C0 Register

Bit	7	6	5	4	3	2	1	0
Name	A2X	—	—	—	—	—	—	—
R/W	R	—	—	—	—	—	—	—
POR	0	—	—	—	—	—	—	—

Bit 7 **A2X**: Operational amplifier output; positive logic. This bit is read only.

Bit 6~0 unimplemented, read as "0"

OPA2C1 Register

Bit	7	6	5	4	3	2	1	0
Name	A2O2CIN	A2O2N	A2PSEL1	A2PSEL0	A2PS	A2NS	A2OEN	A2EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	1	1	0	0

Bit 7 **A2O2CIN**: OPA2 output to comparator input control bit
0: disable
1: enable

Bit 6 **A2O2N**: OPA2 output to OPA2 Inverting input control bit
0: disable
1: enable

Bit 5~4 **A2PSEL1, A2PSEL0**: OPA2 Non-inverting input selection bit
00: no connection
01: from VH1 (0.9×VLDO)
10: from VM (0.5×VDD or 0.5×VLDO)
11: from VL1 (0.1×VDD or 0.1×VLDO)

Bit 3 **A2PS**: A2P pin to OPA2 Non-inverting input control bit
0: no connection
1: from A2P pin

Bit 2 **A2NS**: A2N pin to OPA2 Inverting input control bit
0: no connection
1: from A2N pin

Bit 1 **A2OEN**: OPA2 output enable or disable control bit
0: disable
1: enable

Note: If OPA2 enable and A2OEN set to 1, the MCU will consumption more DC power (100uA ~ 200uA).

Bit 0 **A2EN**: OPA2 enable or disable control bit
0: disable
1: enable

OPA2C2 Register

Bit	7	6	5	4	3	2	1	0
Name	A1O2A2N	A1O2A2P	—	—	PGAEN	PGA2	PGA1	PGA0
R/W	R/W	R/W	—	—	R/W	R/W	R/W	R/W
POR	0	0	—	—	0	0	0	0

Bit 7 **A1O2A2N**: OPA1 output to OPA2 Inverting input control bit
0: disable
1: enable

Bit 6 **A1O2A2P**: OPA1 output to OPA2 Non-inverting input control bit
0: disable
1: enable

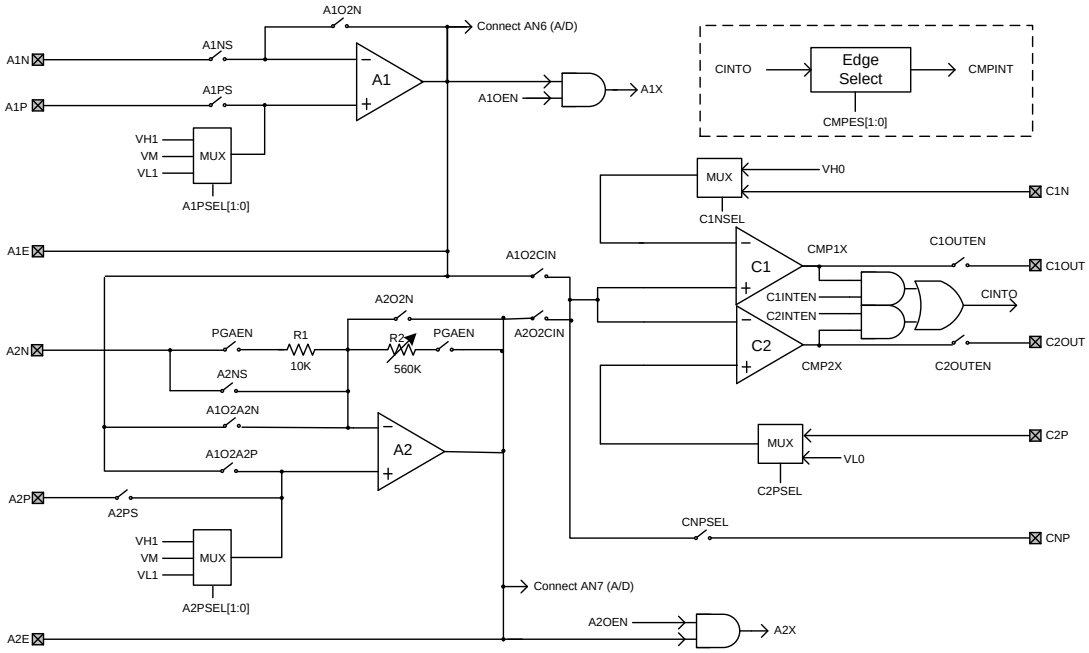
Bit 5~4 unimplemented, read as "0"

Bit 3 **PGAEN**: OPA2 PGA gain enable control bits
0: disable
1: enable

Bit 2~0 **PGA2, PGA1, PGA0**: OPA2 Gain control bits
000: 1
001: 8
010: 16
011: 24
100: 32
101: 40
110: 48
111: 56

Operational Amplifier Operation

The advantages of multiple switches and input path options, various reference voltage selection, up to 8 kinds of internal software gain control, offset reference voltage calibration function and power down control for low power consumption enhance the flexibility of these two OPAs to suit a wide range of application possibilities. The following block diagram illustrates the main functional blocks of the OPAs and Comparator in this device.



Operational Amplifier Functions

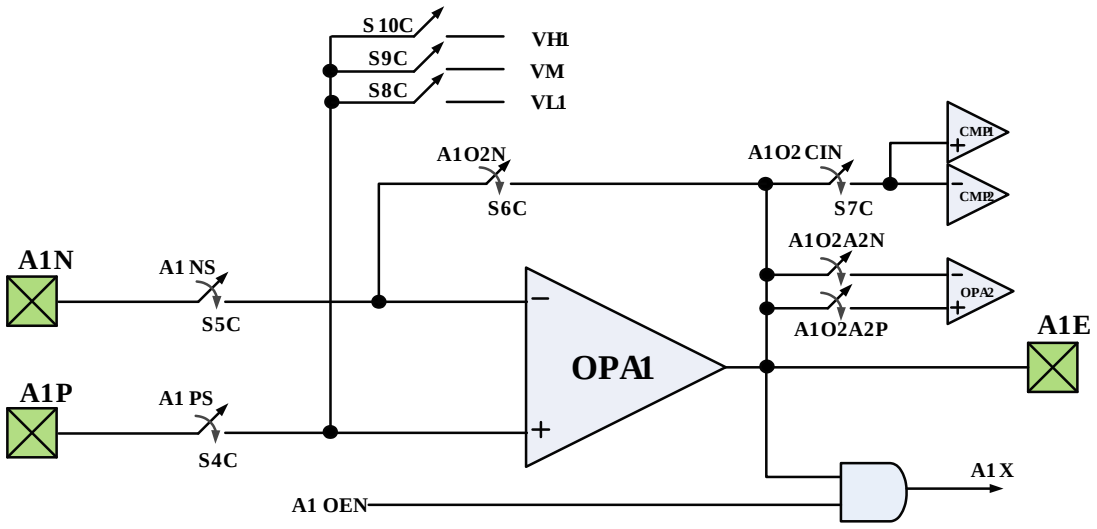
The OPAs are connected together internally in a specific way and the output of OPAs can also be connected to the internal comparators as shown in the block diagram. Each of the OPAs has its own control register, with the name OPA1C0, OPA1C1, OPA2C0, OPA2C1 and OPA2C2 which are used to control the enable/disable function, the calibration procedure and the programmable gain function of OPA2.

OPA1 Switch Control

The following diagram and table illustrate the OPA1 switch control setting and the corresponding connections. Note that some switch control selections will force some switches to be controlled by hardware automatically.

For example:

- The S7C is closed when A1O2CIN=1 and the S7C is opened when A1O2CIN =0.
- The A1PS=1 will force A1PSEL1,0=(00), i.e. S10C,S9C,S8C will be opened.
- When the A1EN=0, S6C switch are opened by hardware, then the related I/O pins can be used as the other functions.



A1PS=1 will force A1PSEL1,0=(00) ,i.e .S 10C, S9C ,S8 C will be opened .

OPA1 switch control:

The following table illustrates the relationship between OPA1 control register settings and the switches:

OPA1 Control Bits in OPA1C0, OPA1C1				Switch Description				Results
A1PS	A1NS	A1PSEL1 A1PSEL0	A1O2N	S4C	S5C	S6C	S8C~ S10C	OPA1 connections
1	1	00	0	On	On	Off	Off	Input= A1N, A1P
0	1	01	0	Off	On	Off	S10C On	Input= A1N, VH1
0	1	10	0	Off	On	Off	S9C On	Input= A1N, VM
0	1	11	0	Off	On	Off	S8C On	Input= A1N, VL1
1	1	00	1	On	On	On	Off	Input= A1N, A1P, connect A1N, A1E
1	0	00	1	On	Off	On	Off	Input= A1P, OPA1 as unit gain buffer
0	1	01	0	Off	On	Off	S10C On	Input= A1N, VH1
0	1	10	0	Off	On	Off	S9C On	Input= A1N, VM
0	1	11	0	Off	On	Off	S8C On	Input= A1N, VL1

OPA1 & I/O status description:

The following table illustrates the OPA1 & I/O settings.

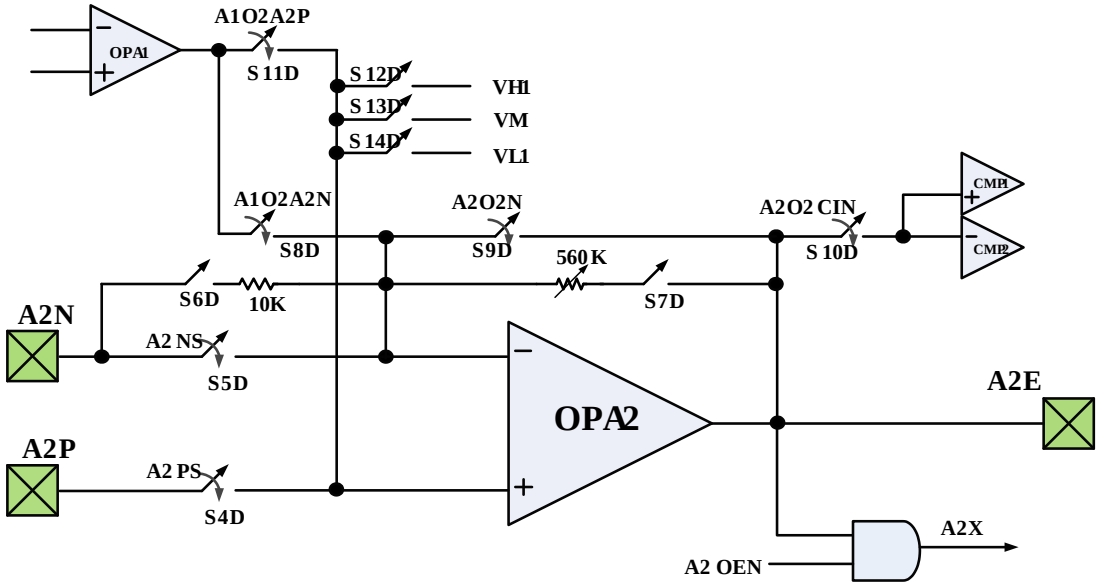
A1EN	A1NS	A1PS	Description
0	x	x	PA2 and PA3 and PA4 are I/Os
1	0	0	PA2 and PA3 are I/Os, PA4 is OPA1 A1E output
1	0	1	PA3 is I/O. PA2 is OPA1 A1P input, PA4 is OPA1 A1E output
1	1	0	PA2 is I/O. PA3 is OPA1 A1N input, PA4 is OPA1 A1E output
1	1	1	PA2 is OPA1 A1P input and PA3 is OPA1 A1N input, PA4 is OPA1 A1E output

OPA2 Switch Control

The following diagram and table illustrate the OPA2 switch control setting and the corresponding connections. Note that some switch control selections will force some switches to be controlled by hardware automatically.

For example:

- The PGAEN=1 will force S6D, S7D to close and the PGAEN=0 will force S6D, S7D to open.
- When the A2EN=0, these switches, S6D, S7D and S9D, are opened by hardware, then the related I/O pins can be used as the other functions.



Switch priority S4D > S11D >(S12D ,S13D ,S14D) ; If A2PS=1 ,S11D~S14D will be opened by hardware .
Switch priority S5D > S8D ; If A2 NS=1 ,S8 D will be opened by hardware .

OPA2 switch control:

The following table illustrates the relationship between OPA2 control register settings and the switches:

OPA2 Control Bits in OPA2C0, OPA2C1, OPA2C2							Switch Description							Results
A2PS	A2NS	A2PSEL1 A2PSEL0	A1O2A2P	A1O2A2N	PGAEN	A2O2N	S4D	S5D	S6/7D	S8D	S9D	S11D	S12D~S14D	OPA2 connections
1	1	00	0	0	0	0	On	On	Off	Off	Off	Off	Off	Normal mode, Input = A2N, A2P
0	1	01	0	0	0	0	Off	On	Off	Off	Off	Off	S12D On	Input = A2N, VH1
0	1	10	0	0	0	0	Off	On	Off	Off	Off	Off	S13D On	Input = A2N, VM
0	1	11	0	0	0	0	Off	On	Off	Off	Off	Off	S14D On	Input = A2N, VL1
0	1	00	1	0	0	0	Off	On	Off	Off	Off	On	Off	Input = A2N, A1E
1	0	00	0	1	0	0	On	Off	Off	On	Off	Off	Off	Input = A1E, A2P
1	1	00	0	0	1	0	On	On	On	Off	Off	Off	Off	Input = A2N, A2P
1	0	00	0	0	1	0	On	Off	On	Off	Off	Off	Off	Input = A2N, A2P
1	0	00	0	0	0	1	On	Off	Off	Off	On	Off	Off	Input = A2P, OPA2 as buffer
0	0	01	0	0	0	1	Off	Off	Off	Off	Off	Off	S12D On	Input = VH1, OPA2 as buffer
0	0	10	0	0	0	1	Off	Off	Off	Off	Off	Off	S13D On	Input = VM, OPA2 as buffer
0	0	11	0	0	0	1	Off	Off	Off	Off	Off	Off	S14D On	Input = VL1, OPA2 as buffer

OPA2 & I/O status description:

The following table illustrates the OPA2 & I/O settings.

A2EN	PGAEN	A2NS	A2PS	Description
0	x	x	x	PA5 and PA6 and PA7 are I/Os
1	0	0	0	PA5 and PA6 are I/Os. PA7 is OPA2 A2E output
1	0	0	1	PA6 is I/O. PA5 is OPA2 A2P input, PA7 is OPA2 A2E output
1	0	1	0	PA5 is I/O. PA6 is OPA2 A2N input, PA7 is OPA2 A2E output
1	0	1	1	PA5 is OPA2 A2P input and PA6 is OPA2 A2N input, PA7 is OPA2 A2E output
1	1	0	0	PA5 is I/O. PA6 is OPA2 A2N input, PA7 is OPA2 A2E output
1	1	0	1	PA5 is OPA2 A2P input and PA6 is OPA2 A2N input, PA7 is OPA2 A2E output
1	1	1	0	PA5 is I/O. PA6 is OPA2 A2N input and bypass R1 (10kΩ), PA7 is OPA2 A2E output
1	1	1	1	PA5 is OPA2 A2P input and PA6 is OPA2 A2N input and bypass R1 (10kΩ), PA7 is OPA2 A2E output

Comparators

Two analog comparators are contained within this device. These functions offer flexibility via their register controlled features such as power-down, interrupt etc. In sharing their pins with normal I/O pins, the comparators do not waste precious I/O pins if there functions are otherwise unused. In addition, the HT45F23A provides the calibration function to adjust the comparator offset.

Comparator Operation

The device contain two comparator functions which are used to compare two analog voltages and provide an output based on their difference. Full control over the two internal comparators is provided via control registers, CMP1C0, CMP1C1, CMP2C0 and CMP2C1. The comparator output is recorded via a bit in their respective control register, but can also be transferred out onto a shared I/O pin or to generate an interrupt trigger with edge control function. Additional comparator functions include the power down control.

Comparator Registers

The internal dual comparators are fully under the control of internal registers, CMP1C0, CMP1C1, CMP2C0 and CMP2C1. These registers control enable/disable function, input path selection, interrupt edge control and input offset voltage calibration function.

CMP1C0 Register

Bit	7	6	5	4	3	2	1	0
Name	CMP1X	C1OFM	C1RS	C1OF4	C1OF3	C1OF2	C1OF1	C1OF0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	1	0	0	0	0

Bit 7 **CMP1X**: comparator output; positive logic. This bit is read only.

Bit 6 **C1OFM**: Comparator mode or input offset voltage cancellation mode
0: comparator mode
1: input offset voltage cancellation mode
When the C1OFM=1, comparator inputs are always from I/O pins. i.e. the CNPSEL and C1NSEL will be forced to "1". That means disconnect the input from OPAs output.

Bit 5 **C1RS**: Comparator input offset voltage cancellation reference selection bit
0: select C1N as the reference input
1: select CNP as the reference input

Bit 4~0 **C1OF4~C1OF0**: Comparator input offset voltage cancellation control bits

CMP1C1 Register

Bit	7	6	5	4	3	2	1	0
Name	CNPSEL	—	—	—	C1INTEN	C1OUTEN	C1NSEL	CMP1EN
R/W	R/W	—	—	—	R/W	R/W	R/W	R/W
POR	1	—	—	—	0	0	1	0

Bit 7 **CNPSEL**: Comparator non-inverting input control bit
0: from OPA output
1: from CNP pin

Bit 6~ 4 unimplemented, read as "0"

Bit 3 **C1INTEN**: Comparator 1 interrupt control bit
0: disable
1: enable

Bit 2 **C1OUTEN**: Comparator 1 output pin control bit
0: disable
1: enable

Bit 1 **C1NSEL**: Comparator 1 inverting input control bit
0: from VH0
1: from C1N pin

Bit 0 **CMP1EN**: Comparator 1 enable or disable control bit
0: disable
1: enable

CMP2C0 Register

Bit	7	6	5	4	3	2	1	0
Name	CMP2X	C2OFM	C2RS	C2OF4	C2OF3	C2OF2	C2OF1	C2OF0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	1	0	0	0	0

Bit 7 **CMP2X**: comparator output; positive logic. This bit is read only.

Bit 6 **C2OFM**: Comparator mode or input offset voltage cancellation mode
0: comparator mode
1: input offset voltage cancellation mode
When the C2OFM=1, comparator inputs are always from I/O pins. i.e. the CNPSEL and C1NSEL will be forced to "1". That means disconnect the input from OPAs output.

Bit 5 **C2RS**: Operational amplifier input offset voltage cancellation reference selection bit
0: select C2P as the reference input
1: select CNP as the reference input

Bit 4~0 **C2OF4~C2OF0**: Comparator input offset voltage cancellation control bits

CMP2C1 Register

Bit	7	6	5	4	3	2	1	0
Name	CMPES1	CMPES0	—	—	C2INTEN	C2OUTEN	C2PSEL	CMP2EN
R/W	R/W	R/W	—	—	R/W	R/W	R/W	R/W
POR	0	0	—	—	0	0	1	0

Bit 7~6 **CMPES1, CMPES0**: Interrupt edge control bits
00: disable
01: rising edge trigger
10: falling edge trigger
11: dual edge trigger

Bit 5~4 unimplemented, read as "0"

Bit 3 **C2INTEN**: Comparator 2 interrupt control bit
0: disable
1: enable

Bit 2 **C2OUTEN**: Comparator 2 output pin control bit
0: disable
1: enable

Bit 1 **C2PSEL**: Comparator 2 non-inverting input control bit
0: from VL0
1: from C2P pin

Bit 0 **CMP2EN**: Comparator 2 enable or disable control bit
0: disable
1: enable

Comparator Functions

Theses two comparators can operate together with the OPAs or standalone as shown in the main functional blocks of the OPAs and Comparators.

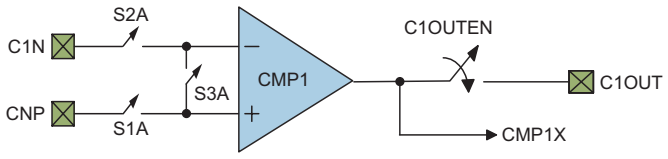
Each of the internal comparators in the HT45F23A allows for a common mode adjustment method of its input offset voltage.

The calibration steps are as following:

- Set C1OFM=1 to setup the offset cancellation mode, here S3A is closed.
- Set C1RS to select which input pin is to be used as the reference voltage - S1A or S2A is closed.
- Adjust C1OF0~C1OF4 until the output status changes.
- Set C1OFM = 0 to restore the normal comparator mode.
- Repeat the same procedure from steps 1 to 4 for comparator 2.

C1OFM	C1RS	S1A	S2A	S3A
0	X	ON	ON	OFF
1	0	OFF	ON	ON
1	1	ON	OFF	ON

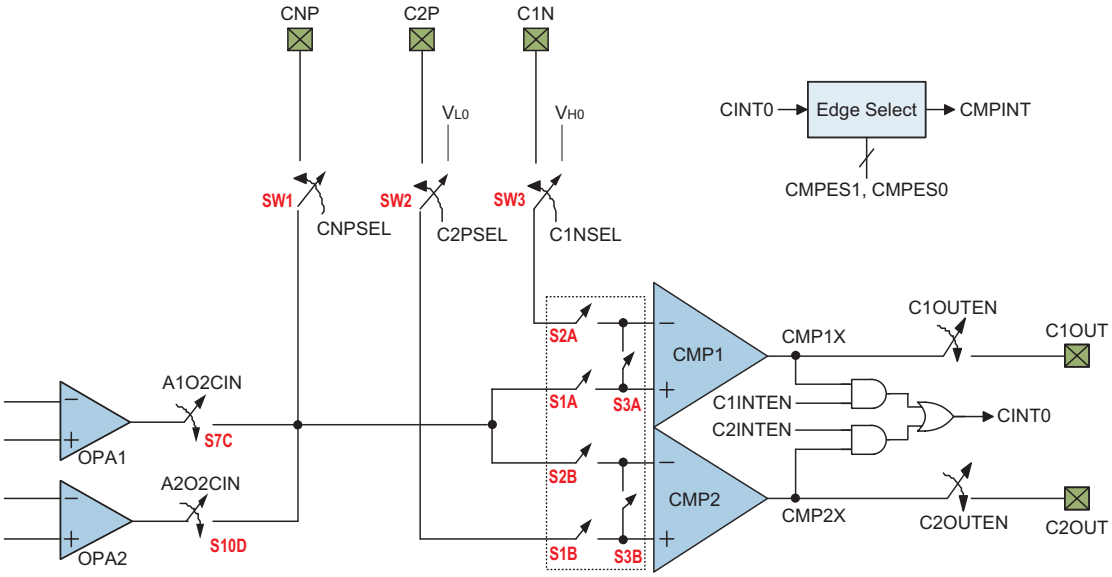
"X": don't care



The following diagram and table illustrate the comparators switch control setting and the corresponding connections. Note that some switch control selections will force some switches to be controlled by hardware automatically.

For example:

- When the CMP1 in calibration mode, i.e. C1OFM =1, then the SW1, SW3 will be forced to close. The CNPSEL and C1NSEL bits will be set "1" by hardware, and these two bits will be read out as "1". After the offset voltage calibration , the CNPSEL and C1NSEL will be back to its original value.
- When the CMP2 in calibration mode, i.e. C2OFM =1, then the SW1, SW2 will be forced to close. The CNPSEL and C2PSEL bits will be set "1" by hardware, and these two bits will be read out as "1". After the offset voltage calibration, the CNPSEL and C2PSEL will be back to its original value.
- If the CNPSEL=1, the A1O2CIN and A2O2CIN will be forced to "0", i.e. If the SWI is closed, and that will force S7C and S10D to open.
- If the CNPSEL=0 and the A1O2CIN=1, the A2O2CIN will be forced to "0", i.e. If the S7C is closed, and that will force S10D to open.



CMP1 & I/O status description:

The following table illustrates the CMP1 & I/O settings.

CMP1EN	C1OUTEN	CNPSEL	C1NSEL	Description
0	x	x	x	PC5 and PA0 and PA1 are I/Os
1	0	0	0	CNP is from OPA1 or OPA2 output, C1N is from VH0 input, PA1 is I/O
1	0	0	1	CNP is from OPA1 or OPA2 output, C1N is from PC5 input, PA1 is I/O
1	0	1	0	CNP is from PA0 input, C1N is from VH0 input, PA1 is I/O
1	0	1	1	CNP is from PA0 input, C1N is from PC5 input, PA1 is I/O
1	1	0	0	CNP is from OPA1 or OPA2 output, C1N is from VH0 input, PA1 is comparator output
1	1	0	1	CNP is from OPA1 or OPA2 output, C1N is from PC5 input, PA1 is comparator output
1	1	1	0	CNP is from PA0 input, C1N is from VH0 input, PA1 is comparator output
1	1	1	1	CNP is from PA0 input, C1N is from PC5 input, PA1 is comparator output

CMP2 & I/O status description:

The following table illustrates the CMP2 & I/O settings.

CMP2EN	C2OUTEN	CNPSEL	C2PSEL	Description
0	x	x	x	PC6 and PA0 and PA2 are I/Os
1	0	0	0	CNP is from OPA1 or OPA2 output, C2P is from VL0 input, PA2 is I/O
1	0	0	1	CNP is from OPA1 or OPA2 output, C2P is from PC6 input, PA2 is I/O
1	0	1	0	CNP is from PA0 input, C2P is from VL0 input, PA2 is I/O
1	0	1	1	CNP is from PA0 input, C2P is from PC6 input, PA2 is I/O
1	1	0	0	CNP is from OPA1 or OPA2 output, C2P is from VL0 input, PA2 is comparator output
1	1	0	1	CNP is from OPA1 or OPA2 output, C2P is from PC6 input, PA2 is comparator output
1	1	1	0	CNP is from PA0 input, C2P is from VL0 input, PA2 is comparator output
1	1	1	1	CNP is from PA0 input, C2P is from PC6 input, PA2 is comparator output

Comparators switch control:

The following table illustrates the relationship between comparators control register settings and the switches:

CMP1,CMP2 Control Bits					Switch Description								Results
CNPSEL	C2PSEL	C1NSEL	C1OFM	C1RS	SW1	SW2	SW3	S1A	S2A	S3A	S7C	S10D	Connections
1 (Forced to 1)	x	1 (Forced to 1)	1	1	ON CNP	x	ON C1N	ON	OFF	ON	OFF	OFF	Input common mode= CNP
1 (Forced to 1)	x	1 (Forced to 1)	1	0	ON CNP	x	ON C1N	OFF	ON	ON	OFF	OFF	Input common mode= C1N
1	0	1	0	x	ON	x	C1N	ON	ON	OFF	OFF	OFF	Input = CNP, C1N
1	0	0	0	x	ON	x	VH	ON	ON	OFF	OFF	OFF	Input = CNP, VH1
0	0	1	0	x	OFF	x	C1N	ON	ON	OFF	ON	OFF	Input = A1E, C1N
0	0	1	0	x	OFF	x	C1N	ON	ON	OFF	OFF	ON	Input = A2E, C1N

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer/Event Counter or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs.The device contains several external interrupt and internal interrupts functions. The external interrupts are controlled by the action of the external INT0, INT1 and $\overline{\text{PINT}}$ pins, while the internal interrupts are controlled by the Timer/Event Counter overflows, the Time Base interrupts, the SPI/I^2C interrupt, the A/D converter interrupt,Comparator interrupt, EEPROM interrupt and LVD interrupt.

Interrupt Register

Overall interrupt control, which means interrupt enabling and request flag setting, is controlled by the INTC0, INTC1, MFIC0 and MFIC1 registers, which are located in the Data Memory. By controlling the appropriate enable bits in these registers each individual interrupt can be enabled or disabled. Also when an interrupt occurs, the corresponding request flag will be set by the microcontroller. The global enable flag if cleared to zero will disable all interrupts.

Interrupt Operation

A Timer/Event Counter overflow, Time Base 0/1, SPI/I²C data transfer complete, an end of A/D conversion, the external interrupt line being triggered, a comparator output, an EEPROM Write or Read cycle ends, or a LVD detection will all generate an interrupt request by setting their corresponding request flag, if their appropriate interrupt enable bit is set. When this happens, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a JMP statement which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a RETI statement, which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagram with their order of priority.

Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded. If an interrupt requires immediate servicing while the program is already in another interrupt service routine, the EMI bit should be set after entering the routine, to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the Stack Pointer is decremented. If immediate service is desired, the stack must be prevented from becoming full.

Interrupt Priority

Interrupts, occurring in the interval between the rising edges of two consecutive T2 pulses, will be serviced on the latter of the two T2 pulses, if the corresponding interrupts are enabled. In case of simultaneous requests, the following table shows the priority that is applied.

Interrupt Source	Priority	Vector
External interrupt 0	1	04H
External interrupt 1	2	08H
Timer/Event Counter 1 overflow	3	0CH
Timer/Event Counter 0 overflow	4	10H
SPI/I²C interrupt	5	14H
Multi-function Interrupt	6	18H

The A/D converter interrupt, Time Base interrupt, External Peripheral interrupt, Comparator interrupt, EEPROM interrupt, and LVD interrupt all share the same interrupt vector which is 18H. Each of these interrupts has their own individual interrupt flag but also share the same MFF interrupt flag. The MFF flag will be cleared by hardware once the Multi-function interrupt is serviced, however the individual interrupts that have triggered the Multifunction interrupt need to be cleared by the application program.

INTC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	T0F	EIF1	EIF0	ET0I	EEI1	EEI0	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 unimplemented, read as "0"

Bit 6 **T0F**: Timer/Event Counter 0 interrupt request flag
0: inactive
1: active

Bit 5 **EIF1**: External interrupt 1 request flag
0: inactive
1: active

Bit 4 **EIF0**: External interrupt 0 request flag
0: inactive
1: active

Bit 3 **ET0I**: Timer/Event Counter 0 interrupt enable
0: disable
1: enable

Bit 2 **EEI1**: External interrupt 1 enable
0: disable
1: enable

Bit 1 **EEI0**: External interrupt 0 enable
0: disable
1: enable

Bit 0 **EMI**: Master interrupt global enable
0: disable
1: enable

INTC1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	MFF	SIMF	T1F	—	EMFI	ESIM	ET1I
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7 unimplemented, read as "0"

Bit 6 **MFF**: Multi-function interrupt request flag
0: inactive
1: active

Bit 5 **SIMF**: SPI/I^2C interrupt request flag
0: inactive
1: active

Bit 4 **TIF**: Timer/Event counter 1 interrupt request flag
0: inactive
1: active

Bit 3 unimplemented, read as "0"

Bit 2 **EMFI**: Multi-function interrupt enable
0: disable
1: enable

Bit 1 **ESIM**: SPI/I^2C interrupt enable
0: disable
1: enable

Bit 0 **ET1I**: Timer/Event counter 1 interrupt enable
0: disable
1: enable

MFIC0 Register

Bit	7	6	5	4	3	2	1	0
Name	PEF	TB1F	TB0F	ADF	EPI	TB1E	TB0E	EADI
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **PEF**: External peripheral interrupt request flag
0: inactive
1: active

Bit 6 **TB1F**: Time Base 1 interrupt request flag
0: inactive
1: active

Bit 5 **TB0F**: Time Base 0 interrupt request flag
0: inactive
1: active

Bit 4 **ADF**: A/D converter interrupt request flag
0: inactive
1: active

Bit 3 **EPI**: External peripheral interrupt enable
0: disable
1: enable

Bit 2 **TB1E**: Time Base 1 enable
0: disable
1: enable

Bit 1 **TB0E**: Time Base 0 enable
0: disable
1: enable

Bit 0 **EADI**: A/D converter interrupt enable
0: disable
1: enable

MFIC1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	LVDF	E2F	CF	—	ELVDI	EE2I	ECI
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7 unimplemented, read as "0"

Bit 6 **LVDF**: LVD interrupt request flag
0: inactive
1: active

Bit 5 **E2F**: EEPROM interrupt request flag
0: inactive
1: active

Bit 4 **CF**: Comparator interrupt request flag
0: inactive
1: active

Bit 3 unimplemented, read as "0"

Bit 2 **ELVDI**: LVD interrupt enable
0: disable
1: enable

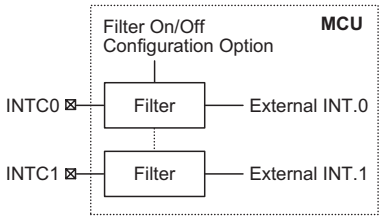
Bit 1 **EE2I**: EEPROM interrupt enable
0: disable
1: enable

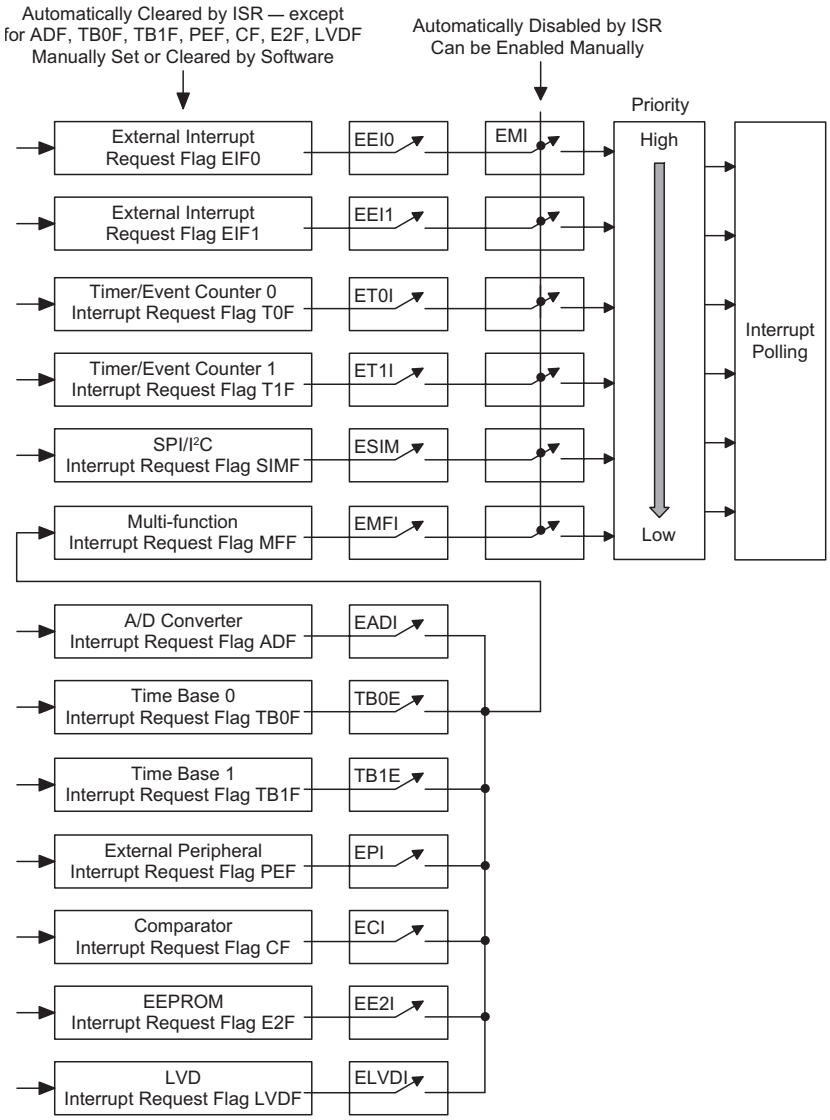
Bit 0 **ECI**: Comparator interrupt enable
0: disable
1: enable

External Interrupt

The external interrupts are controlled by signal transitions on the pins INT0~INT1. An external interrupt request will take place when the external interrupt request flags, EIF0~EIF1, are set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pins. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit, EEI0~EEI1, must first be set. Additionally the correct interrupt edge type must be selected using the INTEDGE register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their external interrupt enable bit in the corresponding interrupt register has been set. The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flags, EIF0~EIF1, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pins will remain valid even if the pin is used as an external interrupt input.

The INTEDGE register is used to select the type of active edge that will trigger the external interrupt. A choice of either rising or falling or both edge types can be chosen to trigger an external interrupt. Note that the INTEDGE register can also be used to disable the external interrupt function.





Interrupt Structure

The external interrupt pins are connected to an internal filter to reduce the possibility of unwanted external interrupts due to adverse noise or spikes on the external interrupt input signal. As this internal filter circuit will consume a limited amount of power, a configuration option is provided to switch off the filter function, an option which may be beneficial in power sensitive applications, but in which the integrity of the input signal is high. Care must be taken when using the filter on/off configuration option as it will be applied not only to both the external interrupt pins but also to the Timer/Event Counter external input pins. Individual external interrupt or Timer/Event Counter pins cannot be selected to have a filter on/off function.

INTEDGE Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 unimplemented, read as "0"

Bit 3~2 **INT1S1, INT1S0**: INT1 Edge select
00: disable
01: rising edge trigger
10: falling edge trigger
11: dual edge trigger

Bit 1~0 **INT0S1, INT0S0**: INT0 Edge select
00: disable
01: rising edge trigger
10: falling edge trigger
11: dual edge trigger

External Peripheral Interrupt

The External Peripheral Interrupt operates in a similar way to the external interrupt and is contained within the Multi-function interrupt.

For an external peripheral interrupt to occur, the global interrupt enable bit, EMI, external peripheral interrupt enable bit, EPI, and Multi-function interrupt enable bit, EMFI, must first be set. An actual external peripheral interrupt will take place when the external interrupt request flag, PEF, is set, a situation that will occur when a negative transition, appears on the $\overline{\text{PINT}}$ pin. The external peripheral interrupt pin is pin-shared with the I/O pin PB5, and is configured as a peripheral interrupt pin via a configuration option. When the interrupt is enabled, the stack is not full and a negative transition type appears on the external peripheral interrupt pin, a subroutine call to the Multi-function interrupt vector at location18H, will take place. When the external peripheral interrupt is serviced, the EMI bit will be cleared to disable other interrupts, however only the MFF interrupt request flag will be reset. As the PEF flag will not be automatically reset, it has to be cleared by the application program.

Timer/Event Counter Interrupt

For a Timer/Event Counter interrupt to occur, the global interrupt enable bit, EMI, and the corresponding timer interrupt enable bit, ET0I or ET1I, must first be set. An actual Timer/Event Counter interrupt will take place when the Timer/Event Counter request flag, T0F or T1F, is set, a situation that will occur when the Timer/Event Counter overflows. When the interrupt is enabled, the stack is not full and a Timer/Event Counter overflow occurs, a subroutine call to the timer interrupt vector at location 0CH or 10H, will take place. When the interrupt is serviced, the timer interrupt request flag, T0F or T1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

SPI/I²C Interface Interrupt

For an SPI/I²C interrupt to occur, the global interrupt enable bit, EMI, and the corresponding interrupt enable bit, ESIM must be first set. An actual SPI/I²C interrupt will take place when the SPI/I²C interface request flag, SIMF, is set, a situation that will occur when a byte of data has been transmitted or received by the SPI/I²C interface When the interrupt is enabled, the stack is not full and a byte of data has been transmitted or received by the SPI/I²C interface, a subroutine call to the SPI/I²C interrupt vector at location 14H, will take place. When the interrupt is serviced, the SPI/I²C request flag, SIMF will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Multi-function Interrupt

An additional interrupt known as the Multi-function interrupt is provided. Unlike the other interrupts, this interrupt has no independent source, but rather is formed from four other existing interrupt sources, namely the A/D Converter interrupt, Time Base interrupts, the External Peripheral interrupt, Comparator interrupt, EEPROM interrupt and LVD interrupt.

For a Multi-function interrupt to occur, the global interrupt enable bit, EMI, and the Multi-function interrupt enable bit, EMFI, must first be set. An actual Multi-function interrupt will take place when the Multi-function interrupt request flag, MFF, is set. This will occur when either a Time Base overflow, an A/D conversion completion, an External Peripheral Interrupt, a Comparator output interrupt, an EEPROM Write or Read cycle ends interrupt, or a LVD interrupt is generated. When the interrupt is enabled and the stack is not full, and either one of the interrupts contained within the Multi-function interrupt occurs, a subroutine call to the Multi-function interrupt vector at location 018H will take place. When the interrupt is serviced, the Multi-Function request flag, MFF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. However, it must be noted that the request flags from the original source of the Multi-function interrupt, namely

the Time-Base interrupt, A/D Converter interrupt or External Peripheral interrupt will not be automatically reset and must be manually reset by the application program.

A/D Interrupt

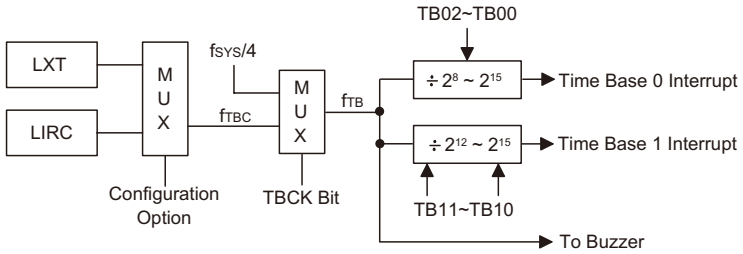
The A/D Interrupt is contained within the Multi-function Interrupt.

For an A/D Interrupt to be generated, the global interrupt enable bit, EMI, A/D Interrupt enable bit, EADI, and Multi-function interrupt enable bit, EMFI, must first be set. An actual A/D Interrupt will take place when the A/D Interrupt request flag, ADF, is set, a situation that will occur when the A/D conversion process has finished. When the interrupt is enabled, the stack is not full and the A/D conversion process has ended, a subroutine call to the Multi-function interrupt vector at location18H, will take place. When the A/D Interrupt is serviced, the EMI bit will be cleared to disable other interrupts, however only the MFF interrupt request flag will be reset. As the ADF flag will not be automatically reset, it has to be cleared by the application program.

Time Base Interrupt

The function of the Time Base Interrupts is to provide regular time signal in the form of an internal interrupt. They are controlled by the overflow signals from their respective timer functions. When these happens their respective interrupt request flags, TB0F or TB1F will be set. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI and Time Base enable bits, TB0E or TB1E, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to their respective vector locations will take place. When the interrupt is serviced, the respective interrupt request flag, TB0F or TB1F, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

The purpose of the Time Base Interrupt is to provide an interrupt signal at fixed time periods. Their clock sources originate from the internal clock source f_{TB}. This f_{TB} input clock passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TBC register to obtain longer interrupt periods whose value ranges. The clock source that generates f_{TB}, which in turn controls the Time Base interrupt period, can originate from several different sources, as shown in the System Operating Mode section.



Time Base Interrupt

TBC Register

Bit	7	6	5	4	3	2	1	0
Name	TBON	TBCK	TB11	TB10	LXTLP	TB02	TB01	TB00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **TBON**: TB0 and TB1 Control
0: disable
1: enable

Bit 6 **TBCK**: Select f_{TB} Clock
0: f_{TBC}
1: $f_{SYS}/4$

Bit 5~4 **TB11~TB10**: Select Time Base 1 Time-out Period
00: $4096/f_{TB}$
01: $8192/f_{TB}$
10: $16384/f_{TB}$
11: $32768/f_{TB}$

Bit 3 **LXTLP**: LXT Low Power Control
0: disable
1: enable

Bit 2~0 **TB02~TB00**: Select Time Base 0 Time-out period
000: $256/f_{TB}$
001: $512/f_{TB}$
010: $1024/f_{TB}$
011: $2048/f_{TB}$
100: $4096/f_{TB}$
101: $8192/f_{TB}$
110: $16384/f_{TB}$
111: $32768/f_{TB}$

Comparator Interrupt

The Comparator Interrupt is contained within the Multi-function Interrupt.

The comparator interrupt is controlled by the two internal comparators. A comparator interrupt request will take place when the comparator interrupt request flag, CF, is set, a situation that will occur when the comparator output changes state. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and comparator interrupt enable bit, ECI , must first be set. When the interrupt is enabled, the stack is not full and the comparator input generates a comparator output transition, a subroutine call to the comparator interrupt vector, will take place. When the Comparator Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the CF flag will not be automatically cleared, it has to be cleared by the application program.

EEPROMInterrupt

The EEPROM Interrupt is contained within the Multi-function Interrupt. An EEPROM Interrupt request will take place when the EEPROM Interrupt request flag, E2F, is set, which occurs when an EEPROM Write or Read cycle ends. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, EEPROM Interrupt enable bit, EE2I, and associated Multi-function interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and an EEPROM Write or Read cycle ends, a subroutine call to the respective Multi-function Interrupt vector, will take place. When the EEPROM Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the E2F flag will not be automatically cleared, it has to be cleared by the application program.

LVD Interrupt

The Low Voltage Detector Interrupt is contained within the Multi-function Interrupt. An LVD Interrupt request will take place when the LVD Interrupt request flag, LVDF, is set, which occurs when the Low Voltage Detector function detects a low power supply voltage. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, Low Voltage Interrupt enable bit, ELVDI, and associated Multi-function interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and a low voltage condition occurs, a subroutine call to the Multi-function Interrupt vector, will take place. When the Low Voltage Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the LVDF flag will not be automatically cleared, it has to be cleared by the application program.

Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the SLEEP or IDLE Mode. A wake-up is generated when an interrupt request flag changes from low to high and is independent of whether the interrupt is enabled or not. Therefore, even though the device is in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt pins, a low power supply voltage or comparator input change may cause their respective interrupt flag to be set high and consequently generate an interrupt. Care must therefore be taken if spurious wake-up situations are to be avoided. If an interrupt wake-up function is to be disabled then the corresponding interrupt request flag should be set high before the device enters the SLEEP or IDLE Mode. The interrupt enable bits have no effect on the interrupt wake-up function.

Programming Considerations

By disabling the interrupt enable bits, a requested interrupt can be prevented from being serviced, however, once an interrupt request flag is set, it will remain in this condition in the INTC0, INTC1, MFIC0 and MFIC1 registers until the corresponding interrupt is serviced or until the request flag is cleared by the application program.

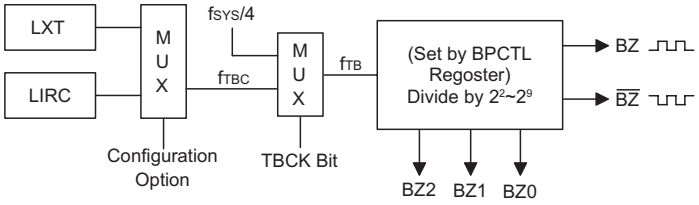
It is recommended that programs do not use the "CALL Subroutine" instruction within the interrupt subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately in some applications. If only one stack is left and the interrupt is not well controlled, the original control sequence will be damaged once a "CALL subroutine" is executed in the interrupt subroutine.

All of these interrupts have the capability of waking up the processor when in the Power Down Mode. Only the Program Counter is pushed onto the stack. If the contents of the status or other registers are altered by the interrupt service program, which may corrupt the desired control sequence, then the contents should be saved in advance.

Buzzer

Operating in a similar way to the Programmable Frequency Divider, the Buzzer function provides a means of producing a variable frequency output, suitable for applications such as Piezo-buzzer driving or other external circuits that require a precise frequency generator. The BZ and $\overline{BZ}$ pins form a complimentary pair, and are pin-shared with I/O pins, PA6 and PA7. A BPCTL register is used to select from one of three buzzer options. The first option is for both pins PA6 and PA7 to be used as normal I/Os, the second option is for both pins to be configured as BZ and $\overline{BZ}$ buzzer pins, the third option selects only the PA6 pin to be used as a BZ buzzer pin with the PA7 pin retaining its normal I/O pin function. Note that the $\overline{BZ}$ pin is the inverse of the BZ pin which together generate a differential output which can supply more power to connected interfaces such as buzzers.

The buzzer is driven by the internal clock source, f_{TB}, which then passes through a divider, the division ratio of which is selected by BPCTL register to provide a range of buzzer frequencies from $f_{TB}/2^2$ to $f_{TB}/2^9$. The clock source that generates f_{TB}, which in turn controls the buzzer frequency, can originate from three different sources, the LXT oscillator, the LIRC oscillator or the System oscillator/4, the choice of which is determined by the f_{TB} clock source option. Note that the buzzer frequency is controlled by BPCTL register, which select the source clock for the internal clock f_{TB}.



Buzzer Function

If the BPCTL options have selected both pins PA6 and PA7 to function as a BZ and $\overline{BZ}$ complementary pair of buzzer outputs, then for correct buzzer operation it is essential that both pins must be setup as outputs by setting bits PAC6 and PAC7 of the PAC port control register to zero. The PA6 data bit in the PA data register must also be set high to enable the buzzer outputs, if set low, both pins PA6 and PA7 will remain low. In this way the single bit PA6 of the PA register can be used as an on/off control for both the BZ and $\overline{BZ}$ buzzer pin outputs. Note that the PA7 data bit in the PA register has no control over the BZ buzzer pin PA7.

BPCTL Register

Bit	7	6	5	4	3	2	1	0
Name	PMODE	PWM0EN	PWM1EN	BC1	BC0	BZ2	BZ1	BZ0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~5 PWM control bits

Bit 4~3 **BC1~BC0**: Buzzer or I/O
- 00: PA7 is I/O, PA6 is I/O
- 01: PA7 is I/O, PA6 is BZ
- 10: reserved
- 11: PA7 is $\overline{BZ}$, PA6 is BZ

Bit 2~0 **BZ2~BZ0**: Buzzer output frequency selection
- 000: $f_{TB}/2^2$
- 001: $f_{TB}/2^3$
- 010: $f_{TB}/2^4$
- 011: $f_{TB}/2^5$
- 100: $f_{TB}/2^6$
- 101: $f_{TB}/2^7$
- 110: $f_{TB}/2^8$
- 111: $f_{TB}/2^9$

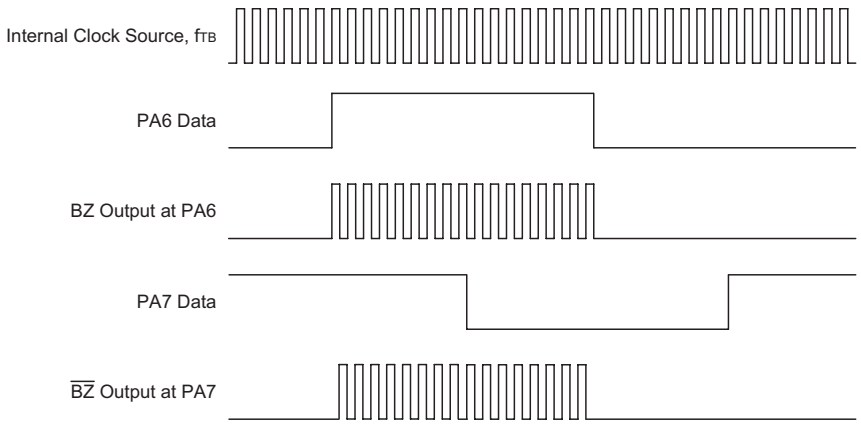
PA6/PA7 Pin Function Control

PAC Register PAC6	PAC Register PAC7	PA Data Register PA6	PA Data Register PA7	Output Function
0	**0**	**1**	**×**	PA6=BZ PA7=$\overline{BZ}$
0	0	0	×	PA6="0" PA7="0"
0	1	1	×	PA6=BZ PA7=input line
0	1	0	×	PA6="0" PA7=input line
1	0	×	D	PA6=input line PA7= D
1	1	×	×	PA6=input line PA7=input line

"x" stands for don't care
"D" stands for Data "0" or "1"

If the options have selected that only the PA6 pin is to function as a BZ buzzer pin, then the PA7 pin can be used as a normal I/O pin. For the PA6 pin to function as a BZ buzzer pin, PA6 must be setup as an output by setting bit PAC6 of the PAC port control register to zero. The PA6 data bit in the PA data register must also be set high to enable the buzzer output, if set low pin PA6 will remain low. In this way the PA6 bit can be used as an on/off control for the BZ buzzer pin PA6. If the PAC6 bit of the PAC port control register is set high, then pin PA6 can still be used as an input even though the option has configured it as a BZ buzzer output.

Note that no matter what BPCTL option is chosen for the buzzer, if the port control register has setup the pin to function as an input, then this will override the BPCTL option selection and force the pin to always behave as an input pin. This arrangement enables the pin to be used as both a buzzer pin and as an input pin, so regardless of the BPCTL option chosen; the actual function of the pin can be changed dynamically by the application program by programming the appropriate port control register bit.



Buzzer Output Pin Control

Note: The above drawing shows the situation where both pins PA6 and PA7 are selected by BPCTL option to be BZ and $\overline{BZ}$ buzzer pin outputs. The Port Control Register of both pins must have already been setup as output. The data setup on pin PA7 has no effect on the buzzer outputs.

Power Down Mode and Wake-up

Entering the IDLE or SLEEP Mode

There is only one way for the device to enter the SLEEP or IDLE Mode and that is to execute the "HALT" instruction in the application program. When this instruction is executed, the following will occur:

- The system clock will be stopped and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting if the WDT clock source is selected to come from the f_{SUB} clock source and the WDT is enabled. The WDT will stop if its clock source originates from the system clock.
- The I/O ports will maintain their present condition.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Standby Current Considerations

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of the device to as low a value as possible, perhaps only in the order of several micro-amps, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised. Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to device which have different package types, as there may be unbonbed pins. These must either be setup as outputs or if setup as inputs must have pull-high resistors connected. Care must also be taken with the loads, which are connected to I/O pins, which are setup as outputs. These should be placed in a condition in which minimum current is drawn or connected only to external circuits that do not draw current, such as other CMOS inputs. Also note that additional standby current will also be required if the configuration options have enabled the LIRC oscillator.

Wake-up

After the system enters the SLEEP or IDLE Mode, it can be woken up from one of various sources listed as follows:

- An external reset
- An external falling edge on Port A
- A system interrupt
- A WDT overflow

If the system is woken up by an external reset, the device will experience a full system reset, however, if the device is woken up by a WDT overflow, a Watchdog Timer reset will be initiated. Although both of these wake-up methods will initiate a reset operation, the actual source of the wake-up can be determined by examining the TO and PDF flags. The PDF flag is cleared by a system power-up or executing the clear Watchdog Timer instructions and is set when executing the "HALT" instruction. The TO flag is set if a WDT time-out occurs, and causes a wake-up that only resets the Program Counter and Stack Pointer, the other flags remain in their original status.

Each pin on Port A can be setup using the PAWU register to permit a negative transition on the pin to wake-up the system. When a Port A pin wake-up occurs, the program will resume execution at the instruction following the "HALT" instruction.

If the system is woken up by an interrupt, then two possible situations may occur. The first is where the related interrupt is disabled or the interrupt is enabled but the stack is full, in which case the program will resume execution at the instruction following the "HALT" instruction. In this situation, the interrupt which woke-up the device will not be immediately serviced, but will rather be serviced later when the related interrupt is finally enabled or when a stack level becomes free. The other situation is where the related interrupt is enabled and the stack is not full, in which case the regular interrupt response takes place. If an interrupt request flag is set high before entering the SLEEP or IDLE Mode, the wake-up function of the related interrupt will be disabled.

Low Voltage Detector – LVD

This device has a Low Voltage Detector function, also known as LVD. This enabled the device to monitor the power supply voltage, VDD, and provide a warning signal should it fall below a certain level. This function may be especially useful in battery applications where the supply voltage will gradually reduce as the battery ages, as it allows an early warning battery low signal to be generated. The Low Voltage Detector also has the capability of generating an interrupt signal.

LVD Register

The Low Voltage Detector function is controlled using a single register with the name LVDC. Three bits in this register, VLVD2~VLVD0, are used to select one of eight fixed voltages below which a low voltage condition will be detemined. A low voltage condition is indicated when the LVDO bit is set. If the LVDO bit is low, this indicates that the VDD voltage is above the preset low voltage value. The LVDEN bit is used to control the overall on/off function of the low voltage detector. Setting the bit high will enable the low voltage detector. Clearing the bit to zero will switch off the internal low voltage detector circuits. As the low voltage detector will consume a certain amount of power, it may be desirable to switch off the circuit when not in use, an important consideration in power sensitive battery powered applications.

LVDC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	LVDO	LVDEN	—	VLVD2	VLVD1	VLVD0
R/W	—	—	R	R/W	—	R/W	R/W	R/W
POR	—	—	0	0	—	0	0	0

Bit 7~6 unimplemented, read as "0"

Bit 5 **LVDO**: LVD Output Flag
0: no Low Voltage Detect
1: Low Voltage Detect

Bit 4 **LVDEN**: Low Voltage Detector Control
0: disable
1: enable

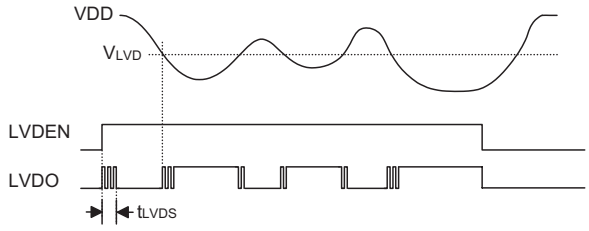
Bit 3 unimplemented, read as "0"

Bit 2~0 **LVD2~LVD0**: Select LVD Voltage
000: 2.0V
001: 2.2V
010: 2.4V
011: 2.7V
100: 3.0V
101: 3.3V
110: 3.6V
111: 4.4V

LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage, V_{DD}, with a pre-specified voltage level stored in the LVDC register. This has a range of between 2.0V and 4.4V. When the power supply voltage, V_{DD}, falls below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. The Low Voltage Detector function is supplied by a reference voltage which will be automatically enabled. When the device is powered down the low voltage detector will remain active if the LVDEN bit is high. After enabling the Low Voltage Detector, a time delay t_{LVDS} should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the V_{DD} voltage may rise and fall rather slowly, at the voltage nears that of V_{LVD}, there may be multiple bit L_{VDO} transitions.

The Low Voltage Detector also has its own interrupt which is contained within one of the Multi-function interrupts, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of t_{LVD} after the LVDO bit has been set high by a low voltage condition. When the device is powered down the Low Voltage Detector will remain active if the LVDEN bit is high. In this case, the LVDF interrupt request flag will be set, causing an interrupt to be generated if VDD falls below the preset LVD voltage. This will cause the device to wake-up from the SLEEP or IDLE Mode, however if the Low Voltage Detector wake up function is not required then the LVDF flag should be first set high before the device enters the SLEEP or IDLE Mode.



LVD Operation

Voice Output

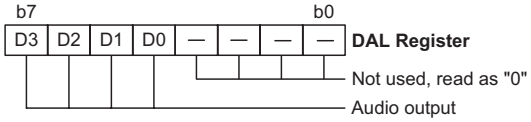
Voice Control

The voice control register controls the DAC circuit. If the DAC circuit is not enabled, any DAH/DAL outputs will be invalid. Writing a "1" to the DACEN bit will enable the DAC circuit and channel the DAC output to its corresponding I/O pin, while writing a "0" to the DACEN bit will disable the DAC circuit.

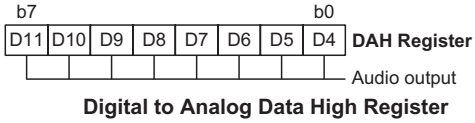
Audio Output and Volume Control – DAL, DAH, DACTRL

The audio output is 12-bits wide whose highest 8-bits are written into the DAH register and whose lowest four bits are written into the highest four bits of the DAL register. Bits 0~3 of the DAL register are always read as zero.

There are 8 levels of volume which are setup using the DACTRL register. The highest 3-bits of this register are used for volume control and the DACEN bit is used to control the DAC function enable or not. Once the DACEN bit is set to "1", this will channel the DAC output to the I/O pin and disable the original I/O pin shared function.



Digital to Analog Data Low Register



Digital to Analog Data High Register

DACTRL Register

Bit	7	6	5	4	3	2	1	0
Name	VOL2	VOL1	VOL0	—	—	—	—	DACEN
R/W	R/W	R/W	R/W	—	—	—	—	R/W
POR	0	0	0	—	—	—	—	0

Bit 7~5 **VOL2~VOL0**: DAC volume control data

Bit 4~1 unimplemented, read as "0"

Bit 0 **DACEN**: DAC enable control bit
0: disable
1: enable

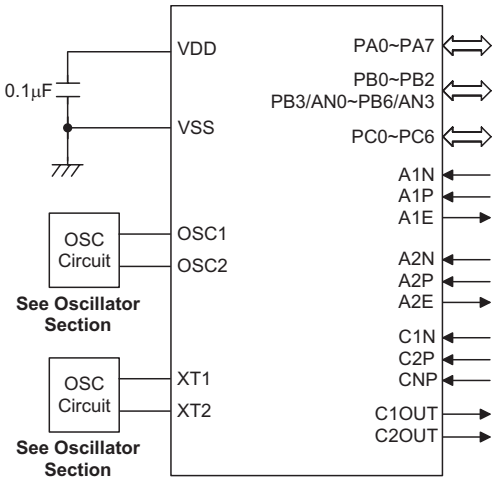
Note: When the DACEN is set to "1", the DAC signal will be channeled to the I/O pin anddisable the original I/O pin shared function.

Configuration Options

Configuration options refer to certain options within the MCU that are programmed into the OTP Program Memory device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. As these options are programmed into the device using the hardware programming tools, once they are selected they can not be changed later by the application software. All options must be defined for proper system function, the details of which are shown in the table.

No.	Options
Oscillator Options	
1	OSC type selection: ERC or crystal or HIRC or EC (external clock) 00. HXT (Filter ON) 01. ERC (Filter ON) 10. HIRC (Filter OFF) 11. EC (Filter OFF)
2	Low speed system oscillator selection-f_L: LXT, LIRC
3	HIRC frequency selection: 4MHz, 910kHz, 2MHz, 8MHz
4	f_S clock selection: f_{SUB} or $f_{SYS}/4$
5	HXT mode selection: 455KHz or 1M~8MHz
Watchdog Options	
6	WDT enable or disable
7	CLRWDT instructions: 1 or 2 instructions
LVR/LVD Options	
8	LVR function: enable or disable
9	LVR voltage : 2.1V or 2.55V or 3.15V or 4.2V
RC Filter	
10	RC filter for TMR0/1 & INT0/1, enable or disable
SPI	
11	SIM enable/disable
12	SPI_WCOL: enable/disable
13	SPI_CSEN: enable/disable, used to enable/disable (1/0) software CSEN function
I^2C	
14	I^2C Debounce Time: no debounce, 1 system clock, 2 system clock
$\overline{RES}$	
15	I/O or $\overline{RES}$ function:
Lock Options	
16	Lock All
17	Partial Lock

Application Circuits



Instruction Set

Introduction

Central to the successful operation of any microcontroller is its instruction set, which is a set of program instruction codes that directs the microcontroller to perform certain operations. In the case of Holtek microcontroller, a comprehensive and flexible set of over 60 instructions is provided to enable programmers to implement their application with the minimum of programming overheads.

For easier understanding of the various instruction codes, they have been subdivided into several functional groupings.

Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5μs and branch or call instructions would be implemented within 1μs. Although instructions which require one more cycle to implement are generally limited to the JMP, CALL, RET, RETI and table read instructions, it is important to realize that any other instructions which involve manipulation of the Program Counter Low register or PCL will also take one more cycle to implement. As instructions which change the contents of the PCL will imply a direct jump to that new address, one more cycle will be required. Examples of such instructions would be "CLR PCL" or "MOV PCL, A". For the case of skip instructions, it must be noted that if the result of the comparison involves a skip operation then this will also take one more cycle, if no skip is involved then only one cycle is required.

Moving and Transferring Data

The transfer of data within the microcontroller program is one of the most frequently used operations. Making use of three kinds of MOV instructions, data can be transferred from registers to the Accumulator and vice-versa as well as being able to move specific immediate data directly into the Accumulator. One of the most important data transfer applications is to receive data from the input ports and transfer data to the output ports.

Arithmetic Operations

The ability to perform certain arithmetic operations and data manipulation is a necessary feature of most microcontroller applications. Within the Holtek microcontroller instruction set are a range of add and subtract instruction mnemonics to enable the necessary arithmetic to be carried out. Care must be taken to ensure correct handling of carry and borrow data when results exceed 255 for addition and less than 0 for subtraction. The increment and decrement instructions INC, INCA, DEC and DECA provide a simple means of increasing or decreasing by a value of one of the values in the destination specified.

Logical and Rotate Operation

The standard logical operations such as AND, OR, XOR and CPL all have their own instruction within the Holtek microcontroller instruction set. As with the case of most instructions involving data manipulation, data must pass through the Accumulator which may involve additional programming steps. In all logical data operations, the zero flag may be set if the result of the operation is zero. Another form of logical data manipulation comes from the rotate instructions such as RR, RL, RRC and RLC which provide a simple means of rotating one bit right or left. Different rotate instructions exist depending on program requirements. Rotate instructions are useful for serial port programming applications where data can be rotated from an internal register into the Carry bit from where it can be examined and the necessary serial bit set high or low. Another application which rotate data operations are used is to implement multiplication and division calculations.

Branches and Control Transfer

Program branching takes the form of either jumps to specified locations using the JMP instruction or to a subroutine using the CALL instruction. They differ in the sense that in the case of a subroutine call, the program must return to the instruction immediately when the subroutine has been carried out. This is done by placing a return instruction "RET" in the subroutine which will cause the program to jump back to the address right after the CALL instruction. In the case of a JMP instruction, the program simply jumps to the desired location. There is no requirement to jump back to the original jumping off point as in the case of the CALL instruction. One special and extremely useful set of branch instructions are the conditional branches. Here a decision is first made regarding the condition of a certain data memory or individual bits. Depending upon the conditions, the program will continue with the next instruction or skip over it and jump to the following instruction. These instructions are the key to decision making and branching within the program perhaps determined by the condition of certain input switches or by the condition of internal data bits.

Bit Operations

The ability to provide single bit operations on Data Memory is an extremely flexible feature of all Holtek microcontrollers. This feature is especially useful for output port bit programming where individual bits or port pins can be directly set high or low using either the "SET [m].i" or "CLR [m].i" instructions respectively. The feature removes the need for programmers to first read the 8-bit output port, manipulate the input data to ensure that other bits are not changed and then output the port with the correct new data. This read-modify-write process is taken care of automatically when these bit operation instructions are used.

Table Read Operations

Data storage is normally implemented by using registers. However, when working with large amounts of fixed data, the volume involved often makes it inconvenient to store the fixed data in the Data Memory. To overcome this problem, Holtek microcontrollers allow an area of Program Memory to be setup as a table where data can be directly stored. A set of easy to use instructions provides the means by which this fixed data can be referenced and retrieved from the Program Memory.

Other Operations

In addition to the above functional instructions, a range of other instructions also exist such as the "HALT" instruction for Power-down operations and instructions to control the operation of the Watchdog Timer for reliable program operations under extreme electric or electromagnetic environments. For their relevant operations, refer to the functional related sections.

Instruction Set Summary

The following table depicts a summary of the instruction set categorised according to function and can be consulted as a basic instruction reference using the following listed conventions.

Table Conventions

x: Bits immediate data
m: Data Memory address
A: Accumulator
i: 0~7 number of bits
addr: Program memory address

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV
ADDM A,[m]	Add ACC to Data Memory	1Note	Z, C, AC, OV
ADD A,x	Add immediate data to ACC	1	Z, C, AC, OV
ADC A,[m]	Add Data Memory to ACC with Carry	1	Z, C, AC, OV
ADCM A,[m]	Add ACC to Data memory with Carry	1Note	Z, C, AC, OV
SUB A,x	Subtract immediate data from the ACC	1	Z, C, AC, OV
SUB A,[m]	Subtract Data Memory from ACC	1	Z, C, AC, OV
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	1Note	Z, C, AC, OV
SBC A,[m]	Subtract Data Memory from ACC with Carry	1	Z, C, AC, OV
SBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	1Note	Z, C, AC, OV
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1Note	C
Logic Operation			
AND A,[m]	Logical AND Data Memory to ACC	1	Z
OR A,[m]	Logical OR Data Memory to ACC	1	Z
XOR A,[m]	Logical XOR Data Memory to ACC	1	Z
ANDM A,[m]	Logical AND ACC to Data Memory	1Note	Z
ORM A,[m]	Logical OR ACC to Data Memory	1Note	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1Note	Z
AND A,x	Logical AND immediate Data to ACC	1	Z
OR A,x	Logical OR immediate Data to ACC	1	Z
XOR A,x	Logical XOR immediate Data to ACC	1	Z
CPL [m]	Complement Data Memory	1Note	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
Increment & Decrement			
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1Note	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1Note	Z
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1Note	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1Note	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1Note	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1Note	C

Mnemonic	Description	Cycles	Flag Affected
Data Move			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1Note	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1Note	None
SET [m].i	Set bit of Data Memory	1Note	None
Branch			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1Note	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1Note	None
SZ [m].i	Skip if bit i of Data Memory is zero	1Note	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1Note	None
SIZ [m]	Skip if increment Data Memory is zero	1Note	None
SDZ [m]	Skip if decrement Data Memory is zero	1Note	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1Note	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1Note	None
CALL addr	Subroutine call	2	None
RET	Return from subroutine	2	None
RET A,x	Return from subroutine and load immediate data to ACC	2	None
RETI	Return from interrupt	2	None
Table Read			
TABRDC [m]	Read table to TBLH and Data Memory	2Note	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2Note	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1Note	None
SET [m]	Set Data Memory	1Note	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
CLR WDT1	Pre-clear Watchdog Timer	1	TO, PDF
CLR WDT2	Pre-clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1Note	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

Note: 1. For skip instructions, if the result of the comparison involves a skip then two cycles are required, if no skip takes place only one cycle is required.

2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.

3. For the "CLR WDT1" and "CLR WDT2" instructions the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after both "CLR WDT1" and "CLR WDT2" instructions are consecutively executed. Otherwise the TO and PDF flags remain unchanged.

Instruction Definition

ADC A,[m] Add Data Memory to ACC with Carry

Description: The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.

Operation: ACC ← ACC + [m] + C

Affected flag(s): OV, Z, AC, C

ADCM A,[m] Add ACC to Data Memory with Carry

Description: The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.

Operation: [m] ← ACC + [m] + C

Affected flag(s): OV, Z, AC, C

ADD A,[m] Add Data Memory to ACC

Description: The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.

Operation: ACC ← ACC + [m]

Affected flag(s): OV, Z, AC, C

ADD A,x Add immediate data to ACC

Description: The contents of the Accumulator and the specified immediate data are added. The result is stored in the Accumulator.

Operation: ACC ← ACC + x

Affected flag(s): OV, Z, AC, C

ADDM A,[m] Add ACC to Data Memory

Description: The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.

Operation: [m] ← ACC + [m]

Affected flag(s): OV, Z, AC, C

AND A,[m] Logical AND Data Memory to ACC

Description: Data in the Accumulator and the specified Data Memory perform a bitwise logical AND operation. The result is stored in the Accumulator.

Operation: ACC ← ACC "AND" [m]

Affected flag(s): Z

AND A,x Logical AND immediate data to ACC

Description: Data in the Accumulator and the specified immediate data perform a bit wise logical AND operation. The result is stored in the Accumulator.

Operation: ACC ← ACC "AND" x

Affected flag(s): Z

ANDM A,[m] Logical AND ACC to Data Memory

Description: Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.

Operation: [m] ← ACC "AND" [m]

Affected flag(s): Z

CALL addr Subroutine call

Description Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.

Operation Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s) None

CLR [m] Clear Data Memory

Description Each bit of the specified Data Memory is cleared to 0.

Operation [m] ← 00H

Affected flag(s) None

CLR [m].i Clear bit of Data Memory

Description Bit i of the specified Data Memory is cleared to 0.

Operation [m].i ← 0

Affected flag(s) None

CLR WDT Clear Watchdog Timer

Description The TO, PDF flags and the WDT are all cleared.

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT1 Pre-clear Watchdog Timer

Description The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT2 and must be executed alternately with CLR WDT2 to have effect. Repetitively executing this instruction without alternately executing CLR WDT2 will have no effect.

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT2 Pre-clear Watchdog Timer

Description The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT1 and must be executed alternately with CLR WDT1 to have effect. Repetitively executing this instruction without alternately executing CLR WDT1 will have no effect.

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CPL [m] Complement Data Memory

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.

Operation [m] ← $\overline{[m]}$

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation $ACC \leftarrow \overline{[m]}$

Affected flag(s) Z

DAA [m] Decimal-Adjust ACC for addition with result in Data Memory

Description Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.

Operation $[m] \leftarrow ACC + 00H$ or
$[m] \leftarrow ACC + 06H$ or
$[m] \leftarrow ACC + 60H$ or
$[m] \leftarrow ACC + 66H$

Affected flag(s) C

DEC [m] Decrement Data Memory

Description Data in the specified Data Memory is decremented by 1.

Operation $[m] \leftarrow [m] - 1$

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

Description Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation $ACC \leftarrow [m] - 1$

Affected flag(s) Z

HALT Enter power down mode

Description This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.

Operation $TO \leftarrow 0$
$PDF \leftarrow 1$

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

Description Data in the specified Data Memory is incremented by 1.

Operation $[m] \leftarrow [m] + 1$

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

Description Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation $ACC \leftarrow [m] + 1$

Affected flag(s) Z

JMP addr Jump unconditionally

Description The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

Description The contents of the specified Data Memory are copied to the Accumulator.

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

Description The contents of the Accumulator are copied to the specified Data Memory.

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

Description No operation is performed. Execution continues with the next instruction.

Operation No operation

Affected flag(s) None

OR A,[m] Logical OR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

Description Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″OR″ x

Affected flag(s) Z

ORM A,[m] Logical OR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

RET Return from subroutine

Description The Program Counter is restored from the stack. Program execution continues at the restored address.

Operation Program Counter ← Stack

Affected flag(s) None

RET A,x Return from subroutine and load immediate data to ACC

Description The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

Description The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← [m].7

Affected flag(s) None

RLA [m] Rotate Data Memory left with result in ACC

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← [m].7

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

Description The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.

Operation [m].(i+1) ← [m].i; (i=0~6)
[m].0 ← C
C ← [m].7

Affected flag(s) C

RLCA [m] Rotate Data Memory left through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← C
C ← [m].7

Affected flag(s) C

RR [m] Rotate Data Memory right

Description The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.

Operation [m].i ← [m].(i+1); (i=0~6)
[m].7 ← [m].0

Affected flag(s) None

RRA [m] Rotate Data Memory right with result in ACC

Description: Data in the specified Data Memory and the carry flag are rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation:
ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← [m].0

Affected flag(s): None

RRC [m] Rotate Data Memory right through Carry

Description: The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.

Operation:
[m].i ← [m].(i+1); (i=0~6)
[m].7 ← C
C ← [m].0

Affected flag(s): C

RRCA [m] Rotate Data Memory right through Carry with result in ACC

Description: Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation:
ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← C
C ← [m].0

Affected flag(s): C

SBC A,[m] Subtract Data Memory from ACC with Carry

Description: The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: ACC ← ACC − [m] − C

Affected flag(s): OV, Z, AC, C

SBCM A,[m] Subtract Data Memory from ACC with Carry and result in Data Memory

Description: The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: [m] ← ACC − [m] − C

Affected flag(s): OV, Z, AC, C

SDZ [m] Skip if decrement Data Memory is 0

Description: The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation:
[m] ← [m] − 1
Skip if [m]=0

Affected flag(s): None

SDZA [m] Skip if decrement Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

SET [m] Set Data Memory

Description Each bit of the specified Data Memory is set to 1.

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

Description Bit i of the specified Data Memory is set to 1.

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

SIZA [m] Skip if increment Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

SNZ [m].i Skip if bit i of Data Memory is not 0

Description If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − [m]

Affected flag(s) OV, Z, AC, C

SUBM A,[m] Subtract Data Memory from ACC with result in Data Memory

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation [m] ← ACC − [m]

Affected flag(s) OV, Z, AC, C

SUB A,x Subtract immediate data from ACC

Description The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − x

Affected flag(s) OV, Z, AC, C

SWAP [m] Swap nibbles of Data Memory

Description The low-order and high-order nibbles of the specified Data Memory are interchanged.

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

SWAPA [m] Swap nibbles of Data Memory with result in ACC

Description The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC.3~ACC.0 ← [m].7~[m].4
ACC.7~ACC.4 ← [m].3~[m].0

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

Description If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation Skip if [m]=0

Affected flag(s) None

SZA [m] Skip if Data Memory is 0 with data movement to ACC

Description The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

SZ [m].i Skip if bit i of Data Memory is 0

Description If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation Skip if [m].i=0

Affected flag(s) None

TABRDC [m] Read table (current page) to TBLH and Data Memory

Description The low byte of the program code (current page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

TABRDL [m] Read table (last page) to TBLH and Data Memory

Description The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

XOR A,[m] Logical XOR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

XORM A,[m] Logical XOR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

Description Data in the Accumulator and the specified immediate data perform a bitwise logical XOR operation. The result is stored in the Accumulator.

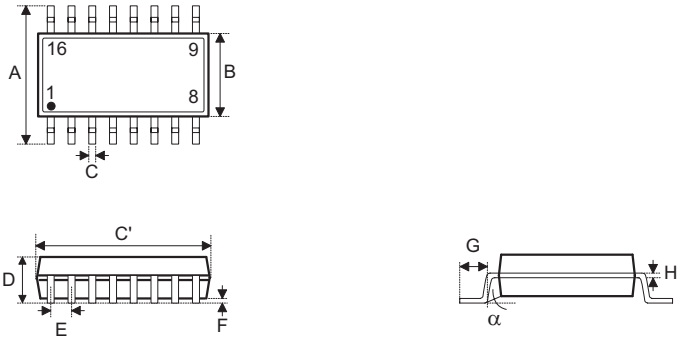
Operation ACC ← ACC ″XOR″ x

Affected flag(s) Z

Package Information

Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the Holtek website (http://www.holtek.com.tw/english/literature/package.pdf) for the latest version of the package information.

16-pin NSOP (150mil) Outline Dimensions

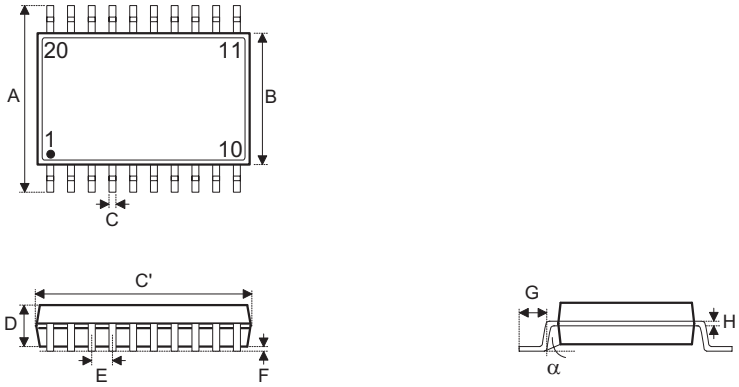


MS-012

Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.228	—	0.244
B	0.150	—	0.157
C	0.012	—	0.020
C′	0.386	—	0.402
D	—	—	0.069
E	—	0.050	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.007	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	5.79	—	6.20
B	3.81	—	3.99
C	0.30	—	0.51
C′	9.80	—	10.21
D	—	—	1.75
E	—	1.27	—
F	0.10	—	0.25
G	0.41	—	1.27
H	0.18	—	0.25
α	0°	—	8°

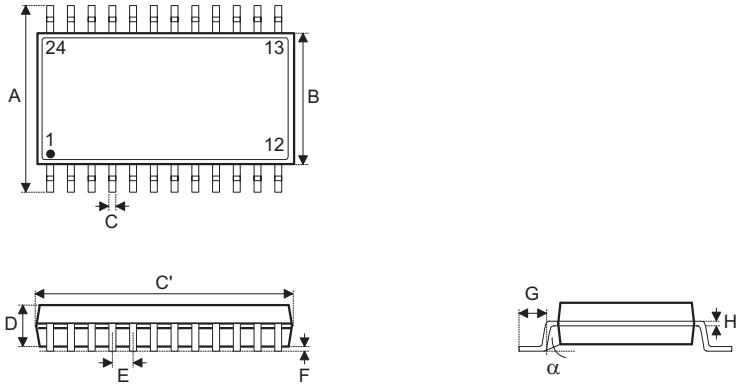
20-pin SSOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.228	—	0.244
B	0.150	—	0.158
C	0.008	—	0.012
C′	0.335	—	0.347
D	0.049	—	0.065
E	—	0.025	—
F	0.004	—	0.010
G	0.015	—	0.050
H	0.007	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	5.79	—	6.20
B	3.81	—	4.01
C	0.20		0.30
C′	8.51	—	8.81
D	1.24	—	1.65
E	—	0.64	—
F	0.10	—	0.25
G	0.38	—	1.27
H	0.18	—	0.25
α	0°	—	8°

24-pin SSOP (150mil) Outline Dimensions

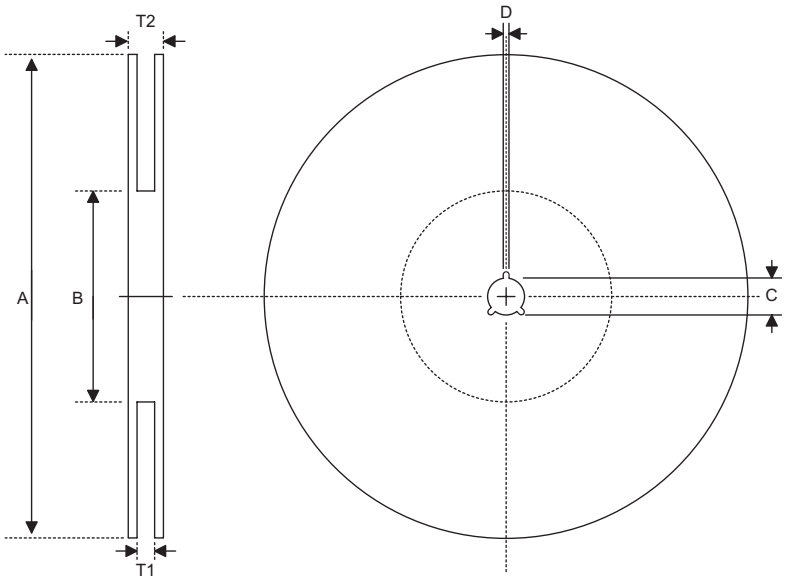


Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.228	—	0.244
B	0.150	—	0.157
C	0.008	—	0.012
C′	0.335	—	0.346
D	0.054	—	0.060
E	—	0.025	—
F	0.004	—	0.010
G	0.022	—	0.028
H	0.007	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	5.79	—	6.20
B	3.81	—	3.99
C	0.20		0.30
C′	8.51	—	8.79
D	1.37	—	1.52
E	—	0.64	—
F	0.10	—	0.25
G	0.56	—	0.71
H	0.18	—	0.25
α	0°	—	8°

Product Tape and Reel Specifications

Reel Dimensions



16-pin NSOP (150mil)

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0±1.0
B	Reel Inner Diameter	100.0±1.5
C	Spindle Hole Diameter	$13.0^{+0.5/-0.2}$
D	Key Slit Width	2.0±0.5
T1	Space Between Flang	$16.8^{+0.3/-0.2}$
T2	Reel Thickness	22.2±0.2

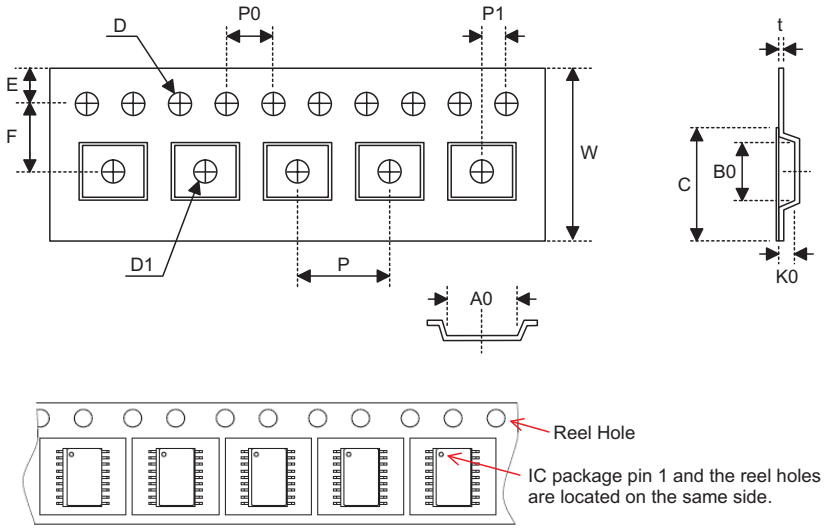
20-pin SSOP (150mil)

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0±1.0
B	Reel Inner Diameter	100.0±1.5
C	Spindle Hole Diameter	$13.0^{+0.5/-0.2}$
D	Key Slit Width	2.0±0.5
T1	Space Between Flang	$16.8^{+0.3/-0.2}$
T2	Reel Thickness	22.2±0.2

24-pin SSOP (150mil)

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0±1.0
B	Reel Inner Diameter	100.0±1.5
C	Spindle Hole Diameter	$13.0^{+0.5/-0.2}$
D	Key Slit Width	2.0±0.5
T1	Space Between Flang	$16.8^{+0.3/-0.2}$
T2	Reel Thickness	22.2±0.2

Carrier Tape Dimensions



16-pin NSOP (150mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	16.0±0.3
P	Cavity Pitch	8.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation(Width Direction)	7.5±0.1
D	Perforation Diameter	$1.55^{+0.10/-0.00}$
D1	Cavity Hole Diameter	$1.50^{+0.25/-0.00}$
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation(Length Direction)	2.0±0.1
A0	Cavity Length	6.5±0.1
B0	Cavity Width	10.3±0.1
K0	Cavity Depth	2.1±0.1
t	Carrier Tape Thickness	0.30±0.05
C	Cover Tape Width	13.3±0.1

20-pin SSOP (150mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	$16.0^{+0.3/-0.1}$
P	Cavity Pitch	8.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation(Width Direction)	7.5±0.1
D	Perforation Diameter	$1.5^{+0.1/-0.0}$
D1	Cavity Hole Diameter	$1.50^{+0.25/-0.00}$
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation(Length Direction)	2.0±0.1
A0	Cavity Length	6.5±0.1
B0	Cavity Width	9.0±0.1
K0	Cavity Depth	2.3±0.1
t	Carrier Tape Thickness	0.30±0.05
C	Cover Tape Width	13.3±0.1

24-pin SSOP (150mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	$16.0^{+0.3/-0.1}$
P	Cavity Pitch	8.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation(Width Direction)	7.5±0.1
D	Perforation Diameter	$1.5^{+0.1/-0.0}$
D1	Cavity Hole Diameter	$1.50^{+0.25/-0.00}$
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation(Length Direction)	2.0±0.1
A0	Cavity Length	6.5±0.1
B0	Cavity Width	9.5±0.1
K0	Cavity Depth	2.1±0.1
t	Carrier Tape Thickness	0.30±0.05
C	Cover Tape Width	13.3±0.1

Holtek Semiconductor Inc. (Headquarters)
No.3, Creation Rd. II, Science Park, Hsinchu, Taiwan
Tel: 886-3-563-1999
Fax: 886-3-563-1189
http://www.holtek.com.tw

Holtek Semiconductor Inc. (Taipei Sales Office)
4F-2, No. 3-2, YuanQu St., Nankang Software Park, Taipei 115, Taiwan
Tel: 886-2-2655-7070
Fax: 886-2-2655-7373
Fax: 886-2-2655-7383 (International sales hotline)

Holtek Semiconductor Inc. (Shenzhen Sales Office)
5F, Unit A, Productivity Building, No.5 Gaoxin M 2nd Road, Nanshan District, Shenzhen, China 518057
Tel: 86-755-8616-9908, 86-755-8616-9308
Fax: 86-755-8616-9722

Holtek Semiconductor (USA), Inc. (North America Sales Office)
46729 Fremont Blvd., Fremont, CA 94538, USA
Tel: 1-510-252-9880
Fax: 1-510-252-9885
http://www.holtek.com